Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Nodo EthMesh, enlace entre una red inalámbrica de respaldo y una red local Ethernet para el sistema anticolisiones Acumine.

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura

Esteban Camacho Sánchez

Cartago, Junio de 2011

INSTITUTO TECNOLOGICO DE COSTA RICA
ESCUELA DE INGENIERIA ELECTRONICA

PROYECTO DE GRADUACIÓN

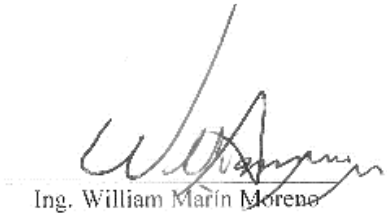
TRIBUNAL EVALUADOR

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Electrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

Ing. Johan Carvajal Godínez
Profesor lector

Ing. Aníbal Coto Cortés
Profesor lector



Ing. William Marín Moreno
Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Escuela de Ingeniería Electrónica

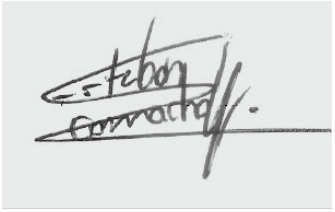
Cartago, Costa Rica. 06 jun. 11

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Cartago, 30 de mayo 2011.



Esteban Camacho Sánchez

Cédula: 3-0396-0238

Resumen

En la implementación de tecnologías actuales, se hace cada vez más indispensable el uso de dispositivos que tengan la capacidad de comunicarse de manera eficaz. La comunicación eficaz se define de muchas formas, una forma de hacerlo es evaluando las facilidades que la misma le brinda a un grupo de usuarios. El crecimiento de una red implica satisfacer ciertas necesidades, como la mezcla entre distintas tecnologías de comunicación para formar una sola red donde los usuarios pueden "brincar de una red a otra. Es entonces un compromiso de parte de los diseñadores e investigadores en comunicaciones proponer soluciones para realizar el enlace de distintas redes y optimizar el aprovechamiento de los recursos disponibles en un entorno.

En este proyecto se entablan las bases para realizar un enlace entre una tecnología de comunicación cableada muy conocida (Ethernet) y una red inalámbrica personalizada, que brinda soporte a otra red en una frecuencia más alta y que tiene características más complejas (red mesh). La idea es aprovechar la red de respaldo para brindar a los usuarios acceso a cierta información de una manera útil y escalable.

En el capítulo uno se define el entorno y la necesidad basándose en un proceso de investigación, aprendizaje y observación. Se identifican las funciones de la red en interés y se plantea cómo la información aprovechable en esa banda de frecuencias puede ser usada. Luego en se concibe a una solución a través del diseño y construcción de un dispositivo, cuyas funciones y alcances quedan descritos por los objetivos planteados en el capítulo segundo.

Posteriormente en los capítulos cuarto y quinto se ilustra una descripción detallada del hardware involucrado y del software contenido en el mismo, así como todas las herramientas de prueba y metodología para su desarrollo.

Posteriormente en el sexto capítulo se corrobora el funcionamiento deseado con algunas mediciones que son comparadas con lo esperado y con algunos conceptos teóricos. Finalmente en el último capítulo se resaltan algunas conclusiones y recomendaciones para que otros puedan realizar mejoras al sistema implementado.

Abstract

Designing of new technology is related to more efficient communication between devices. An effective system of communications is defined on many ways; one important parameter could be the quality of access for a communications network. The growth of a network means satisfaction of some needs as combination of different technologies of communication to create a homogeneous network. Because for that, designers and researchers have the compromise to propose integrated solutions to connect different networks and optimize available resource´s use.

This project establishes fundamentals for connecting a LAN network with a sensor wireless network. This wireless network has support functions for a higher frequency network (mesh). The principal idea in this project is to use the support network for bringing useful information to some users.

At chapter one, we investigate and observe to describe the project´s environment and understand the problematic. We identified the network´s most important information that could be useful to control processes. Then solution is described in as the design and construction of one link device, the device has the capability of connect both networks and route the information between one Pc and one RF node, goals and limitations are described in the second chapter.

Then, chapter 4 and 5 explains the hardware and software in detail; also describe the methodology and tools used during development and design of this project. At the sixth chapter, this document presents some measures and compares them against theory to verify the results. Finally in last chapter, there are some conclusions and recommendations for anyone want to optimize the Ethmesh project.

Dedicatoria y Agradecimiento

Dedicado a la mujer que ha hecho toda mi carrera posible, que siempre creyó en mí y me dio todo su apoyo sin dudarlo ni un segundo. Gracias a mi grandiosa madre.

Por otro lado agradezco muchísimo al Dr.Pablo Mandolesi y Dr. Fabio Manson que me dieron la autorización y guía para poder trabajar en la prestigiosa Universidad Nacional del Sur.

Agradezco a los chicos del grupo de investigación por su asesoría y amistad, gracias a los ingenieros Sebastian Armano, Martín Ceci, José Moyano y Lisandro Pérez.

Un reconocimiento enorme a Hernán Gutiérrez por su gran ayuda en el diseño y depuración del hardware de este proyecto.

Un agradecimiento especial al Dr. Alfonso Chacón quien fue actor fundamental para abrirme paso hacia la UNS, confiando en mi y recomendado a mi persona ante sus colegas.

Por último un agradecimiento a la Vice-rectoría de Vida Estudiantil y de Servicios Académicos (VIESA), por su apoyo financiero para mi viaje.

INDICE GENERAL

Índice de Figuras

Índice de Tablas

Capítulo 1: Introducción

1.1 Entorno del Proyecto

El Instituto de Investigaciones en Ingeniería Eléctrica (IIIE) Alfredo Desages, nació como iniciativa del personal docente y becarios del Departamento de Ingeniería Eléctrica y de Computadores (DIEC). Este ente de investigación lleva cabo sus actividades en las instalaciones de La Universidad Nacional del Sur (UNS), ubicada en Bahía Blanca Buenos Aires, provincia de Argentina. Al momento que comenzó con sus actividades estaba compuesto por cinco grupos: Sistemas Digitales, Sistemas de Programación, Comunicaciones, Dinámica de Sistemas y Control. Durante los primeros años se desarrollaron distintos proyectos relacionados a redes de sensores, éstos proporcionaron un marco de trabajo a los estudiantes del DIEC para realizar sus proyectos de grado. En un principio los diseños se enfocaron en consideraciones como el consumo de potencia y tamaño de los nodos que conformarían una red, sin embargo conforme las investigaciones se enfocaron a distintas aplicaciones, los objetivos fueron amoldándose a distintas necesidades [1].

El desarrollo de proyectos en redes de sensores requiere de muchas áreas de la electrónica como las comunicaciones eléctricas, la aplicación y diseño de sistemas embebidos e inclusive el diseño VLSI, debido a que la manera en que se implementa una red depende de muchas consideraciones, entre ellas están el comportamiento de los sensores, la arquitectura de los circuitos, las condiciones ambientales de la ubicación de los dispositivos y la tecnología de comunicación que utilizan para enlazarse. El grupo de comunicaciones, conformado por becarios de posgrado de la UNS, es el grupo que ha tenido mayor experiencia en diseños para redes de sensores. Actualmente el grupo trabaja en un proyecto de supervisión vehicular para una empresa de minería a cielo abierto, el proyecto lleva como nombre “sistema anticolisiones AcuMine”, y consiste en colocar dispositivos electrónicos de medición e interfaz en los vehículos pesados e instrumentación mecánica de una mina ubicada en Chile, éstos han sido apodados como “cajas”. Las cajas de AcuMine son un hardware que realiza varias tareas de comunicación, ubicación y medición en los vehículos. Además cuentan con una serie de dispositivos para interfaz humana como pantallas táctiles, alarmas y parlantes para dar avisos.

Las cajas en cada vehículo están conectadas a una red de área de control (CANBus), por la cual viaja toda la información necesaria para las distintas unidades electrónicas de control en el vehículo. Este bus de datos es muy común en automóviles y otros vehículos debido a que cuenta con una alta inmunidad a la interferencia electromagnética [2].

Las cajas utilizan dos bandas de radiofrecuencia para su comunicación. En la banda de mayor frecuencia a 2.4GHz, se manejan los datos "en grueso", esto quiere decir que es el canal para la comunicación entre todos los nodos. Los datos en grueso es la información sobre las ubicaciones anteriores de los nodos, actualizaciones, información recolectada del CANBus, y otros. Para mover esta información con el mayor alcance posible es necesario que los paquetes cuenten con encabezados correspondientes a un protocolo de comunicación OLSR. La topología utilizada es una combinación de una red de infraestructura con una red AD Hoc, este tipo de redes son conocidas como redes Mesh. El esquema de enrutamiento en una red de este tipo demanda algunos requisitos para el mantenimiento de la red, ya que al ser dinámica algunos nodos estarán moviéndose fuera y dentro de red. Es ahí donde está la funcionalidad de la otra banda.

La otra banda tiene su frecuencia central en los 433MHz, en esta banda todas las cajas emiten mensajes para el mantenimiento de la Mesh, éstos son conocidos como mensajes "ping", y es como decir "soy el nodo A y estoy conectado en la posición x,y". Estos mensajes son de suma importancia porque mantienen informados a los otros nodos sobre la presencia de vecinos y por ende sobre posibles rutas para la información. Las ubicaciones de los miembros en la red son obtenidas por un sistema GPS incorporado en cada caja Acumine, y permiten estimar las posiciones relativas entre un nodo y otro, calcular distancias y otros parámetros que en conjunto conforman el sistema anti colisiones.

Los vehículos utilizados en la mina tiene dimisiones de hasta 5 metros de alto, lo que impide al conductor tener visibilidad de toda la periferia del móvil, como solución a esto los investigadores del IIIE han diseñado las cajas para que éstas recopilen información

sobre los distintos eventos que ocurren en el vehículo, así como las posiciones de los demás móviles en la mina, de esta manera pueden asistir al conductor con dificultades como la proximidad de otros camiones, lo cual es muy importante debido a la poca visibilidad ocasionada por el exceso de polvo. De ahí que cada vehículo posea un dispositivo que puede registrar si el vehículo es detenido, si se pone el embrague, si es puesto en reversa o avanza hacia adelante y la velocidad y revoluciones con que éste lo hace. Entonces, como además conoce la posición de todos los otros vehículos móviles en el área de excavación, almacenaje y descarga; se convierte en un efectivo asistente de manejo y sistema de localización.

Como se muestra en la figura 1.1 cada vehículo es visto como un nodo y cada nodo tiene la capacidad de direccionar los datos provenientes de otro, esto es característico de una red Mesh. La ventaja en este tipo de topología, es que un nodo lejano puede comunicarse con un nodo concentrador a pesar de no estar al alcance del mismo, y es precisamente porque lo hace a través de otros nodos en la red. En este caso un vehículo que se encuentre dentro del rango para un punto de acceso, podría actuar como "puente" para que los datos de otro vehículo puedan llegar a su destino. Además tiene como principal característica que es tolerante a fallos, puesto que la caída de un solo nodo no implica la caída del resto de la red [3].

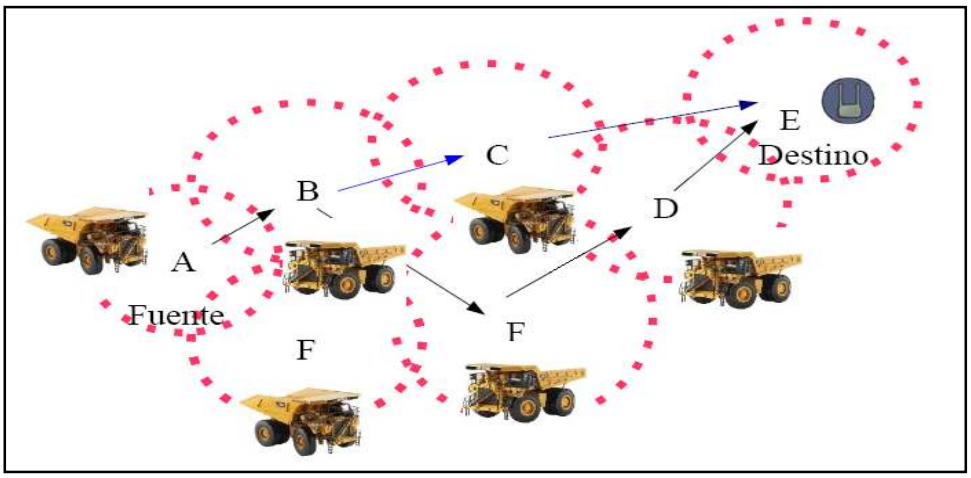


Figura 1.1 Red MESH formada por los vehículos.

1.2 Generalidades

El equipo del IIIE actualmente trabaja en una nueva versión de dispositivo de ubicación y comunicación, esta nueva unidad se ha apodado como "nodo liviano". El nodo liviano pretende ser usado por vehículos invitados que ingresen al área de trabajo, de manera que todo visitante se incorpora a la red al momento de su llegada y puede ser localizado y advertido de las ubicaciones de los demás vehículos. En este punto es donde adquiere más importancia el mantenimiento de la red Mesh, ya que con la implementación de esta nueva unidad se espera que haya mayor incorporación y movimiento de nodos en la red. Lo cual conlleva a dar mayor importancia a la banda de 433MHz como canal de soporte.

En este nuevo modelo se pretende incorporar un transductor con mejores características, el CC1101. Por otro lado las cajas instaladas en los camiones utilizan el transductor CC1000, una versión anterior al CC1101, por ende surge la necesidad de mantener compatibilidad entre los nodos livianos y las cajas instaladas, esto limita el uso del nuevo transductor CC1101. De forma que no es posible utilizar otras modulaciones o tasas de transferencia, sin embargo para esta aplicación en particular, no es de interés utilizar la máxima tasa de transferencia posible (500Kbaudios), esto porque la cantidad de datos que circulan por los 433MHz no es masiva, lo cual es normal en una banda de respaldo [6]. En especial se le da prioridad a la distancia, de forma que los vehículos puedan tener mayor libertad de circulación y mantenerse siempre enlazados.

Otro requisito es que el tamaño de los paquetes en 433MHz pueda ser variable, de manera que quede abierta la posibilidad de agrandar o achicar las tramas dependiendo del tipo de mensajes que se deseen intercambiar entre camiones y nodos livianos. Entonces es importante que cualquier dispositivo que se vaya a incorporar a la red inalámbrica, tenga la capacidad de variar el tamaño de los paquetes, a fin de que se entienda con el resto de nodos de la red. En los apéndices A.1 y A.2 pueden apreciarse unas fotografías de ambos dispositivos, como se puede notar el nodo liviano es una versión mucho más pequeña y sencilla que la caja, además éste no se encuentra conectado al CANBus del vehículo, ya

que su única finalidad es de ubicación y no de asistencia de navegación, porque está diseñado para automóviles o camionetas en las que ingresan visitantes a la mina.

Anteriormente se ha descrito que existen dos tipos de nodo que deben mantener una compatibilidad de comunicación, y que para esa comunicación se utiliza una red Mesh en 2.4GHz y una red más sencilla de mantenimiento en 433MHz. En este punto es conveniente aclarar que algunos dispositivos mecánicos de la mina sufren desgaste debido a las condiciones tan extremas del entorno de trabajo, por ende algunas compuertas, y otros pueden emitir mensajes de alerta sobre fallos o mal funcionamiento a los vehículos a través de la red de 433MHz. Dichos mensajes son "escuchados" por los vehículos que en ese momento se encuentren en el rango de alcance del nodo emisor colocado en la estructura mecánica, activando una alarma visual en la pantalla principal del dispositivo.

Al llegar a este punto de la descripción del entorno, se puede notar que no se cuenta con una forma de monitorizar los mensajes de alerta o de ping que se distribuyen a través de la mina. De manera que si por ejemplo un pala mecánica, presenta un mal funcionamiento el reporte se transmite a los conductores de los camiones, o a los que se encuentren visitando la mina en un vehículo más liviano. Esto quiere decir que son los choferes quienes se dan cuenta de los fallos y quienes deben reportar los fallos, en este sentido se podría considerar como una necesidad contar con alguna forma de notificar esto a una central de control y mantenimiento, ahorrando tiempo y esfuerzo por parte de los conductores de los camiones sin distraerlos de su trabajo, y evitando así que miembros del personal realicen tareas que no les corresponden.

Por otro lado, también sería muy provechoso que se pudieran aprovechar los mensajes ping, que dan mantenimiento a la red, para registrar la presencia de los algunos vehículos en zonas de interés. Por ejemplo controlando la llegada de los camiones que transportan el material de donde se extrae el cobre a la zona de descarga, se podría tener un mayor dominio sobre los procesos de producción en la planta. El ejemplo anterior la da una funcionalidad doble a la banda de 433MHz.

1.3 Síntesis del Problema

A pesar de contar con un sistema de comunicación bien definido entre los miembros circulantes y estáticos de la mina, no se cuenta con una forma de aprovechar los mensajes en la banda de 433MHz, para hacer de forma eficiente los reportes de alerta, ni tampoco para obtener información que podría ser muy provechosa a fin de optimizar procesos de producción y seguridad en el espacio de trabajo.

1.4 Solución Seleccionada

Para solventar la necesidad antes descrita, se propone el diseño de un nodo que facilite el servicio de enlace y que será colocado en algún punto de interés dentro de una limitada distancia a la estación de control. Por ejemplo, el nodo podría ser colocado cerca de la zona de descarga, lugar donde todos los camiones depositan cerca de 5 toneladas de materia prima para extraer el cobre, y de esta forma capturar el mensaje ping emitido cada cierto tiempo por los camiones, y como dentro de este mensaje yace información sobre la ubicación del camión, un supervisor puede darse cuenta cuando cada camión se acerca, llega o se aleja a la zona de descarga sin necesidad de contacto visual. Además como cada camión tiene un número identificador se podría utilizar la información recopilada, para compilar una base de datos sobre la cantidad de descargas diarias de cada vehículo y utilizar estadísticas para el control de procesos en la mina.

La otra aplicación sería utilizar la banda de 433MHz para reportar las fallas hacia la estación de control, en ese caso se utilizaría un espacio dentro del encabezado de la trama para identificarlos como mensajes de alerta, y dentro del campo de datos del paquete podría estar contenida la información detallada de la unidad que está dando problemas. Como la red en 433MHz no tiene el alcance que tiene la Mesh en 2.4GHz y como no se quiere sobrecargar más la red con mensajes de alerta, la idea es aprovechar la movilidad mecánica de los camiones para darle una movilidad a la información dentro de la banda de 433MHz. Esto quiere decir, que cuando un camión se encuentre alejado de la estación de control y recibe un mensaje de alerta a través de la banda de respaldo, éste interpreta el mensaje y lo almacena para retransmitirlo en el momento que se encuentre dentro del alcance de la estación de control, la caja dentro del camión puede percibir cuando está suficientemente cerca debido a que cuenta con un sistema de GPS. Entonces se podría decir que se le asigna una funcionalidad más tanto a la banda de 433MHz como al GPS del Acumine, esto se traduce en un mayor aprovechamiento de un hardware que ya está disponible.

Se puede entonces afirmar que se trabaja bajo el concepto de nodo "EthMesh", que consiste en un nodo de enrutamiento para paquetes de información con tamaño variable, entre una red inalámbrica a frecuencia de 433MHz y una red local Ethernet. Su función principal es tomar la información de distintos nodos RF que circulan en su cercanía y enviarlos hacia una estación central de supervisión donde, por medio de una red Ethernet varios usuarios pueden acceder a la información. Además permite utilizar la banda de mantenimiento de la Mesh para enviar información a cualquier camión en cercanía a la estación, permitiendo así que una banda cuyo objetivo original es el mantenimiento para una red ubicada en una frecuencia más alta, se use para abrir un trecho de comunicación entre un vehículo en las cercanías de la estación y un usuario de control sentado frente a un computador en la estación. La idea de utilizar una red local de computadores para la distribución de la información a todos los supervisores en una estación central fue propuesta por los encargados del proyecto, y la idea de implementar un dispositivo que se incorpore a la red y preste el servicio de enlace es parte del planteamiento de este proyecto.

Otra característica importante del dispositivo a construir es la de poder ser configurado para manejar paquetes de tamaño variable, esto con el fin de hacer el diseño más flexible, de modo que el nodo de enlace podría ser utilizado eventualmente en otras redes. Dejando abierta la posibilidad de que el EthMesh tenga utilidad en nuevas aplicaciones, donde el tamaño de los paquetes puede ser diferente. Por ejemplo en aplicaciones para las cuales el ahorro en el consumo de potencia sea crítico (lo que no pasa en este caso en especial debido a que las cajas están conectadas al sistema eléctrico de los camiones de dónde pueden obtener su alimentación), sería entonces necesario que el tamaño de los paquetes se mantenga mínimo, de forma que el consumo de potencia en las transmisiones por radio sea disminuido.

Considerando que se pretende llevar a cabo un diseño bastante flexible, es importante que varios parámetros que definen la etapa de comunicación por radiofrecuencia puedan ser configurables. Como el dispositivo debe contar con acceso remoto, dicha configuración debe ser realizada a través de la interfaz de comunicación del nodo con la estación central de manera que la configuración para el dispositivo puede llevarse a cabo por cualquiera de los usuarios conectados a la red Ethernet desde la estación central de supervisión.

Para la comunicación por radio frecuencia se utilizará el transductor CC1101, será necesario generar todo el software para hacer un driver del transductor, y éste tendrá que ser compatible con el dispositivo que se elija para manejar la comunicación en el desarrollo de este proyecto. La configuración por defecto del EthMesh deberá ser aquella que garantice la compatibilidad entre este dispositivo a diseñar y los nodos actuales en la mina. Por ende debe estar configurado por defecto para usar una codificación Manchester, una frecuencia central de 433MHz, modulación FSK y cumplir con otras condiciones de manejo digital de la información antes de la modulación para el caso de transmitir y luego de la demodulación en caso de recibir.

Para la implementación de este proyecto será necesario el uso de un dispositivo programable con la capacidad de manejar las dos interfaces de comunicación presentes en el desarrollo de este proyecto, por un lado la interfaz Ethernet, la cual se utilizaría para la distribución de la información en la estación central, y por el otro lado la comunicación inalámbrica, con la que se capturarían los datos generados en un camión o las alertas que deben llegar hasta el personal pertinente. Luego de hacer algunas consideraciones se decidió usar un microcontrolador de la marca Microchip, específicamente el modelo PIC32MXX75F256L.

En la figura 1.2 se ilustra la comunicación resultante entre la estación de control y los vehículos después de la implementación del nodo EthMesh.

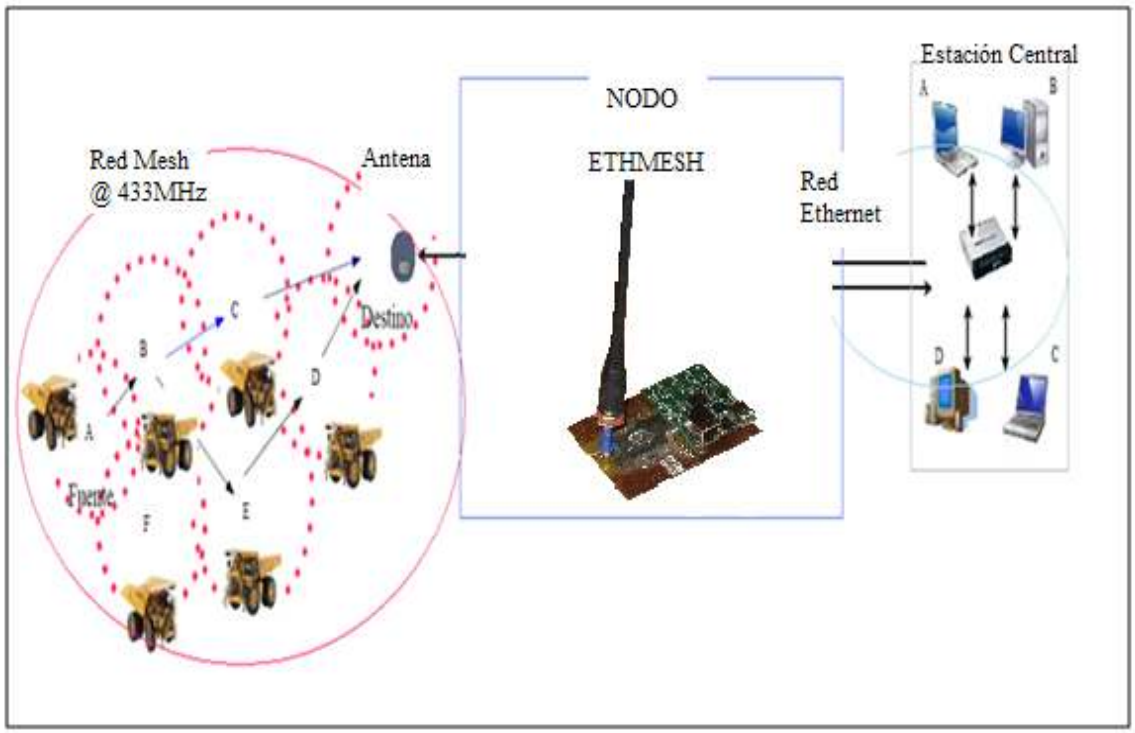


Figura 1.2 Diagrama de las redes entrelazadas a través del EthMesh.

Como se aprecia en la figura 1.2, cuando un vehículo se acerque a la estación de control, cerca de donde se ubica el nodo EthMesh, éste transmite mensajes de posición o de alerta al nodo y a su vez, el EthMesh los retransmite hacia la estación de control por medio de una interfaz de Ethernet.

Lo que se tiene hasta el momento es una red inalámbrica de topología MESH que utiliza una banda de más baja frecuencia para el envío de mensajes hola o ping que le dan mantenimiento a la red y para emitir alertas. Por otro lado en la estación de control se tiene una red Ethernet, la idea es aprovechar esa red local y dirigir hacia ella todos los mensajes de alerta y de ping que emiten los camiones cuando se acercan a la estación de control.

Capítulo 2: Meta y Objetivos

2.1 Meta

Desarrollar herramientas electrónicas que permitan llevar a cabo procedimientos automatizados de supervisión y recopilar información que permita implementar metodologías estocásticas de optimización de procesos de producción.

2.2 Objetivo General

Diseñar e implementar un nodo de servicio de enlace, que tenga la capacidad de realizar el enrutamiento de paquetes de datos de tamaño variable, entre una red inalámbrica de respaldo a una red MESH y una red local Ethernet.

2.3 Objetivos Específicos

1) Utilizar los mismos esquemas de codificación (Manchester) y modulación (FSK@433MHz) para mantener la compatibilidad con la red inalámbrica de respaldo actual, de manera que a pesar de utilizar un controlador de RF diferente al de ciertas unidades de la red, se mantenga la comunicación con todos los nodos.

2) Diseñar el hardware necesario para acoplarlo a una tarjeta de desarrollo de diseños comercial, de manera que ésta adquiera la capacidad de comunicarse por radiofrecuencia con la red inalámbrica de interés, integrando una etapa de RF sin que se presenten interferencias debidas a la radiofrecuencia en el funcionamiento de la etapa digital.

3) Diseñar el programa necesario para el microcontrolador de manera que éste sea capaz de administrar el intercambio de mensajes, entre una computadora representada por una dirección IP y un nodo asociado a una dirección dentro de la red inalámbrica.

4) Desarrollar un programa compatible con el microcontrolador para llevar a cabo el manejo de las comunicaciones por radiofrecuencia a través del transductor CC1101.

5) Diseñar un dispositivo flexible, en el sentido de que éste pueda configurar su etapa de radio, siguiendo las instrucciones que reciba por medio de la interfaz Ethernet, permitiéndole comunicarse dentro de redes inalámbricas con diferentes parámetros.

Capítulo3: Marco Teórico

3.1 Principios de comunicación digital

3.1.1 Modulación Digital

La modulación digital encuentra sus fundamentos en su antecesor la modulación analógica, y esto es así debido a que en realidad son muy similares. Podría decirse que la diferencia más transcendental se encuentra en la forma que lleva la señal de información al momento de ser introducida al modulador. Aunque algunos de los esquemas de modulación digital no se usan para sistemas analógicos, en la mayoría de los casos la única diferencia entre ambos está en que la señal de información es digitalizada y codificada antes de ser modulada. Similar a la modulación analógica, en la modulación digital existe una señal de información y una señal portadora que por lo general tiene una frecuencia mayor a la frecuencia de la señal de información, tal y como se da en los sistemas heterodinos. En la siguiente figura puede apreciarse una versión simplificada de ambos esquemas.

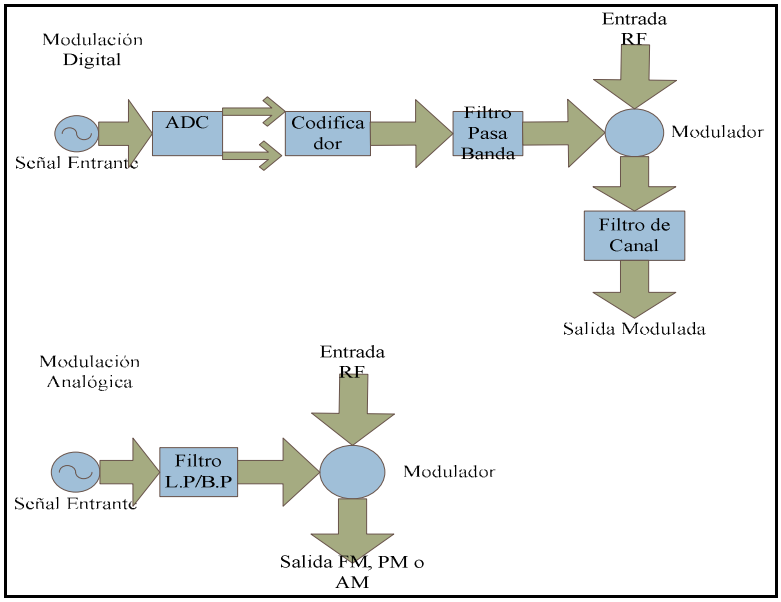


Figura 3.1 Esquemas de modulación digital y analógica.

Como se puede notar en la figura anterior, en la modulación analógica la señal de información primero pasa a un filtro paso bajo o paso banda, y luego es aplicada al modulador para producir la señal FM, AM o PM a transmitir. Por otro lado, en la modulación digital la señal de información es primeramente digitalizada, la cantidad de bits que representan la señal está definida por la resolución deseada para la conversión. Esta información es codificada bajo alguna estructura de codificación que produce la señal digital, ésta señal pasa por un filtro paso banda antes de ser usada para modular la señal portadora.

La modulación digital ofrece mejor eficiencia en el uso del ancho de banda, debido a que se tiene la capacidad de ajuste de la información dentro de un ancho de banda limitado. Es más eficiente en el consumo de potencia, representa menores costos de implementación en comparación con la tecnología analógica y es cuasi libre de perturbaciones en transmisiones de larga distancia.

3.1.4 Modulación por desplazamiento de frecuencia (FSK)

El concepto de modulación por desplazamiento en la frecuencia de una señal portadora para producir una señal de información, es muy similar al de la modulación PSK. La diferencia es que en este caso la señal digital que imprime su información en la modulada, provoca variaciones en la frecuencia, tal y como se muestra en la figura 3.2.

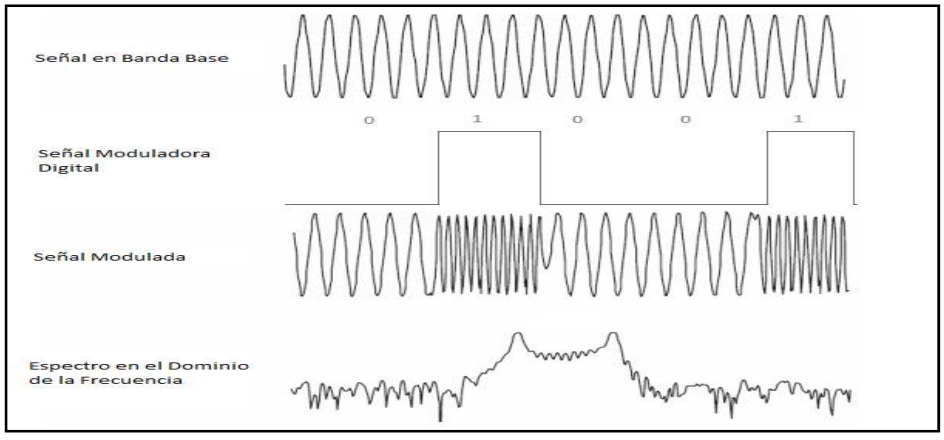


Figura 3.2 Señales involucradas en 2FSK.

Similar al caso de PSK, la forma más sencilla de modular una señal portadora es desplazándola entre dos valores de frecuencia. Esto quiere decir que a partir de una frecuencia central, se suma o se resta una determinada cantidad de frecuencia; esta magnitud es conocida como la desviación de frecuencia y es un importante parámetro ya que determina el ancho de banda ocupado por la señal modulada. Esta modulación es conocida como 2FSK o BFSK, que significa modulación por desplazamiento de frecuencia binaria.

Cuando se disminuye una determinada cantidad de frecuencia, se obtiene como resultado la llamada frecuencia de espacio, mientras que al aumentar la frecuencia se obtiene la frecuencia de marca. Las frecuencias de espacio y de marca son usualmente asignadas a un cero y un uno lógico, en ese orden respectivo, aunque no necesariamente tiene que ser así, ya que bien podría estar asociada la frecuencia más alta a un cero lógico y la más baja a un uno lógico. En la figura 3.2 se muestra el caso más común, donde la frecuencia más baja es de espacio; ya que representa un nivel cero, y la frecuencia más alta es de marca; ya que se encuentra ligada al nivel uno. Por otro lado, en la figura 3.2 también se platea un bosquejo de la forma que puede tener el espectro de la señal resultante modulada.

Haciendo un poco de matemática, se puede decir que para una portadora con forma sinusenoidal y de amplitud **A**, la cantidad de potencia promedio es:

$$\boldsymbol{P_s = \frac{1}{2}A^2\ [W]} \tag{3.1}$$

Por lo tanto, se podría representar la amplitud de la señal portadora modulada como:

$$\boldsymbol{A = \sqrt{2P_s}\ [V]} \tag{3.2}$$

Obteniendo finalmente una representación para la señal modulada como sigue:

$$\boldsymbol{v_{BFSK}(t) = \sqrt{2P_s}\cos[\omega_0 t + d(t)\Omega t]} \tag{3.3}$$

Donde **d(*t*)** puede ser -1 ó +1 en correspondencia con los niveles lógicos 1 ó 0, obteniendo así una señal que puede ser cualquiera de las dos siguientes

$$\boldsymbol{v_{BFSK}(t) = \sqrt{2P_s}\cos(\omega_0 + \Omega)t} \quad (3.4)$$

$$\boldsymbol{v_{BFSK}(t) = \sqrt{2P_s}\cos(\omega_0 - \Omega)t} \quad (3.5)$$

En las ecuaciones anteriores Ω representa la desviación en frecuencia, podría entonces denominarse a $\omega_H = \omega_0 + \Omega$, como la frecuencia de marca y $\omega_L = \omega_0 - \Omega$, como la de espacio. Esta denominación está relacionada a la figura 3.3, donde se plantea la posibilidad de llevar a cabo la modulación 2FSK a partir de la suma de las salidas correspondientes a dos moduladores balanceados, uno de ellos con $\omega_H$como portadora marca y otra con $\omega_L$como portadora espacio.

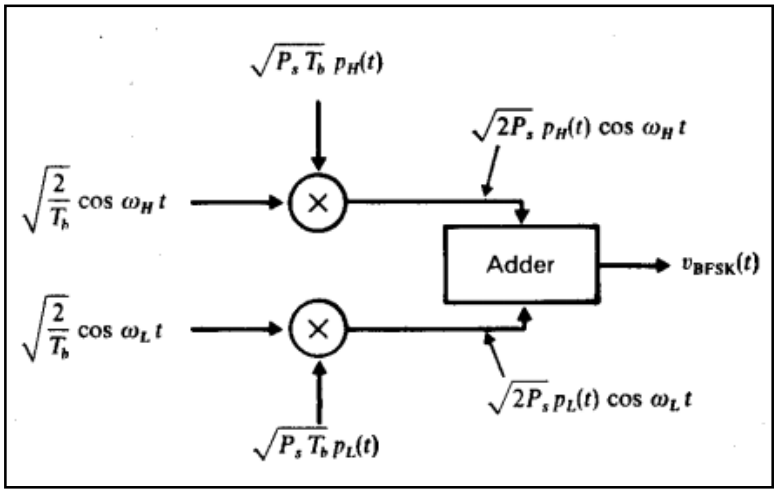


Figura 3.3 Modulador BFSK ó 2FSK. Tomado de [8].

Las representaciones $P_{H(t)}$ y$P_{L(t)}$ están relacionadas al cambio entre 1 ó 0 que se da para asignar cada valor lógico, lo que quiere decir que son funciones opuestas y mientras una vale uno la otra se anula. Produciendo así una función a la salida donde un término es nulo mientras existe el otro y viceversa, la idea se muestra en la tabla 3.1.

Tabla 3.1 Representación de la moduladora.

d(t)	$P_{H(t)}$	$P_{L(t)}$
+1V	+1V	0V
-1V	0V	+1V

En términos de $P_{H(t)}$ y $P_{L(t)}$, la señal BFSK es

$$v_{BFSK} = \sqrt{\frac{P_S}{2}}\cos(\omega_H t + \theta_H) + \sqrt{\frac{P_S}{2}}\cos(\omega_L t + \theta_L) + \sqrt{\frac{P_S}{2}}P_H{}'\cos(\omega_H t + \theta_H) + \sqrt{\frac{P_S}{2}}P_L{}'\cos(\omega_L t + \theta_L)$$

La ecuación 3.6 se ha expresado en términos de $P_H{}'$ y $P_L{}'$, los cuales son bipolares, pues varían entre -1 y +1, y no entre 1 y 0 como $P_{H(t)}$ y $P_{L(t)}$. Este ajuste matemático tiene como único fin el de justificar la forma del espectro que se presenta en la figura 3.8, los primeros dos término de la ecuación 3.6 producen una densidad espectral de potencia que consiste en dos impulso, uno en la frecuencia de espacio y otro en la frecuencia de marca. Mientras que los últimos dos términos producen el espectro de dos señales PSK binarias, como el representado en la figura 3.4.

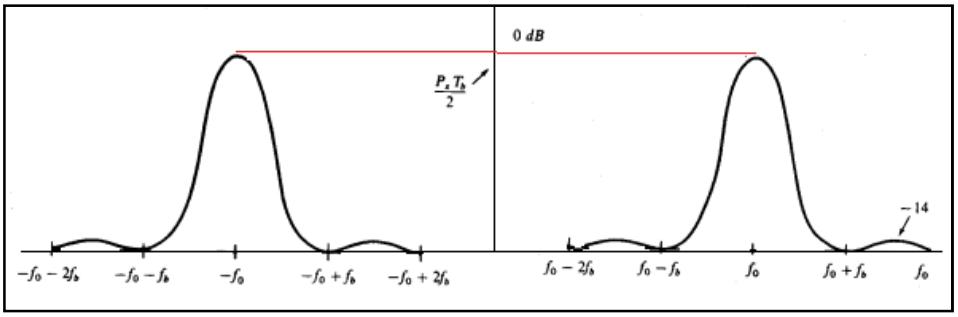


Figura 3.4 Espectro de una señal modulada BPSK. Tomado de [8].

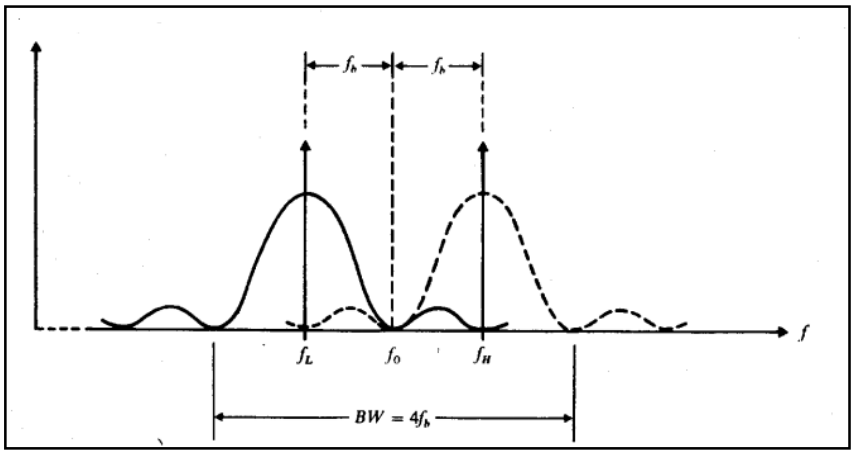


Figura 3.5 Espectro de una señal modulada 2FSK. Tomado de [8].

En la figura 3.5 se ha asumido el caso en que la frecuencia de marca menos la frecuencia de espacio resulta igual a el doble de la frecuencia con la que cambian los bits de la señal modulante, o dicho de otra forma la frecuencia de bit de la señal de información. En términos matemáticos, y llamando a la frecuencia de marca como f_Hy a la frecuencia de espacio como f_L, y además siendo f_b la frecuencia de bit; se tiene entonces la siguiente condición:

$$f_H - f_L = 2f_b \tag{3.6}$$

Bajo esta condición se presenta un traslape entre las dos partes del espectro que permite distinguir los niveles de la señal binaria **d(*t*)**, de manera que se puede estimar el ancho de banda ocupado por esta señal como aproximadamente cuatro veces la frecuencia de bit que tenga la señal moduladora, y el espectro se encuentra ubicado alrededor de la frecuencia central a partir de la cual se generan las desviaciones para imprimir la información [8].

$$\boldsymbol{BW}(BFSK) = 4f_b \text{ [Hz]} \tag{3.7}$$

Tal y como se ilustra en la figura 3.5, el ancho de banda se puede obtener con la ecuación 3.7.

3.2 Redes Ad-Hoc y el esquema de enrutamiento AODV

3.2.1 Concepto de una red Ad Hoc

Una red inalámbrica de topología Ad-hoc es aquella en la que cada nodo está preparado para enviar los paquetes de información a los demás, esto quiere decir que las rutas por donde viaja la información no tienen una forma constante, lo que le da una característica dinámica a la red. El protocolo que utiliza debe facilitar el encadenamiento sin utilizar puntos de acceso, esto quiere decir que no depende de un nodo en especial que gestiona las comunicaciones con el resto de los nodos de la red [10].

3.2.2 Esquema AODV

Para llevar a cabo la implementación de una topología es necesario un esquema de enrutamiento, en este caso se usa AODV, en inglés Ad hoc On-Demand Distance Vector, lo cual quiere decir que el enrutamiento de los datos se realiza bajo demanda. Es una variante del DV [11]. El esquema AODV resuelve algunos problemas de cuentas al infinito que se presentan en DV. En un esquema Ad hoc, toda la red se encuentra en estado de espera hasta el momento en que algún nodo solicita conexión, es entonces cuando éste emite una solicitud por multidifusión que es pasada por varios nodos, cuando estos nodos reciben un mensaje que no es para ellos y lo retransmiten, graban la dirección del nodo que originalmente hizo la solicitud de conexión. De esta manera se crea una colección de diversas rutas temporales de regreso para un mensaje destinado al nodo que acaba de solicitar conexión, las rutas a pesar de ser diversas no son todas usadas, se descartan varias considerando la cantidad de nodos que involucran para retornar un mensaje al destino. Las rutas no utilizadas son borradas de las tablas de enrutamiento luego de un determinado tiempo, y en caso de un error de enrutamiento un mensaje de alerta es pasado al nodo emisor, provocando que el proceso se repita.

Ad hoc bajo demanda pretende disminuir la cantidad de mensajes retransmitidos por la red necesarios para que ésta se mantenga en funcionamiento, para ello utiliza protocolos basados en el uso de números de secuencia para evitar repetir solicitudes de rutas ya conocidas, cantidades limitadas de "saltos" sobre la red con el fin de no saturarla y algoritmos de espera para solicitudes de retransmisión en caso de detectar errores. Sin embargo, este esquema presenta desventajas durante la inicialización de una nueva ruta, es más lenta y produce mayor carga sobre la red que otros esquemas [12].

3.3 Redes Mesh y esquema de enrutamiento OSLR.

3.3.1 Concepto de Red Mesh

Una red mesh es una red inalámbrica con una topología que resulta de la combinación de una red Ad-Hoc con una red Infraestructura. En una red de este tipo se tienen puntos de acceso, lo cual es típico de una red infraestructura y contrario al caso de una Ad-Hoc; de modo que la información debe ser reportada a un nodo que juega el papel de punto de acceso. Sin embargo, la manera en que la información es movida a través de la red es tal y como se hace en una Ad-Hoc, ya que no todos los nodos de la red disponen de un enlace directo con un nodo punto de acceso. Esta combinación de ambas técnicas es lo que se conoce como una red mesh. Como se muestra en la figura 3.6, la topología mesh de un área definida por una zona dentro del alcance de un punto de acceso es conocida como una nube mesh. El acceso a estas nubes depende de la red formada por estos puntos de acceso.

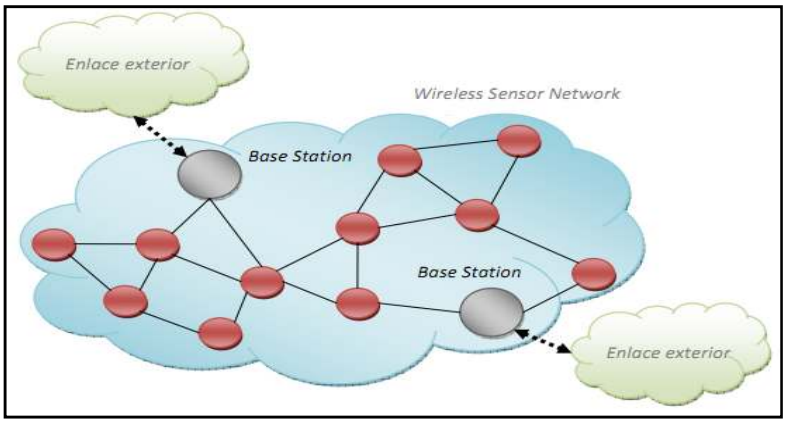


Figura 3.6 Concepto de una red Mesh. Tomado de [15].

Fuera de los nodos conocidos como “estaciones base” o “nodos sumideros”, la red mesh no dispone de ninguna infraestructura. El resto de los nodos dentro de las nubes forman una red ad hoc para transmitir a través de saltos los múltiples paquetes hasta que alcanzan un nodo sumidero. De manera que todos los nodos pueden jugar el papel de host o de enrutador de manera indistinta. Es así como este tipo de red permite unirse a la red a

dispositivos que aunque se encuentran fuera de la cobertura de las estaciones base, están dentro del rango de cobertura de algún nodo en la nube. [15]

3.3.2 Ventajas y desventajas de una red mesh

En una red mesh como los nodos tienen libertad de movilidad éstos se pueden aproximar, aumentando considerablemente la potencia de recepción y haciendo más confiables los enlaces. Además se dice que es auto-reparable ya que si existen espacios sin cobertura pueden agregarse más nodos para reparar esos agujeros. Por ende la caída de un solo nodo no implica la caída de toda la red completa ya que los nodos son capaces de realizar enrutamiento.

En una mesh la redundancia está definida en función de la cantidad de nodos en un espacio determinado (densidad de nodos), la redundancia en un sistema de comunicación se puede agrandar simplemente agregando mayor cantidad de nodos. Una red mesh es también escalable y puede manejar cientos o miles de nodos, ya que no depende de un punto central de control, por el contrario al agregar nodos se obtiene redundancia.

En la siguiente tabla se realiza una comparación entre las diferentes topologías disponibles.

Tabla 3.2 Comparación para aplicaciones industriales. Tomado de [16]

Topología	Confiabilidad	Adaptabilidad	Escalabilidad
Punto a punto	Alta	Baja	Ninguna (dos puntos finales)
Punto a multipunto	Baja	Baja	Moderada (7-30 puntos finales)
Mesh	Alta	Alta	Alta (miles de puntos)

A diferencia de las redes punto a punto, las redes punto multipunto ofrecen mayor escalabilidad, pero su confiabilidad está determinada en función de la ubicación del punto de acceso y el punto final. Por el contrario las redes mesh son capaces de proporcionar

confiabilidad inherente, ya que son adaptables a espacios de múltiples arquitecturas y condiciones ambientales. [16]

3.3.3 Mecanismos de selección de ruta

El uso de una topología mesh para una red de sensores implica la inclusión de mecanismos para transmitir la información desde zonas alejadas hasta un destino que se pretende alcanzar por medio de saltos. Esto conlleva a incluir esquemas de enrutamiento encargados de crear las rutas que permitan retransmitir los paquetes procedentes de los distintos puntos de la red hacia su destino. Los estándares 802.15.4 y 802.11 incluyen el uso de topologías mesh como parte de su implementación sin embargo, no definen los algoritmos para la selección de las rutas en una red de este tipo [17]. Las primeras propuestas para la selección de los vecinos en redes multi-salto utilizan como criterio la calidad de la señal recibida por los nodos cercanos o la proximidad de los vecinos. Una de las dificultades más desafiantes es la movilidad de algunos nodos de la red, lo cual exige que los mecanismos de encaminamiento sean capaces de percibir y adaptarse a cambios topológicos en la red.

Dependiendo del punto de vista del que se analice, un esquema de enrutamiento puede clasificarse en distintas categorías. Si se analizan con base en el mecanismo utilizado para mover los paquetes hasta su destino, entonces se tienen otras tres calcificaciones. Los proactivos crean las rutas previamente a ser requeridas para mover la información. Los reactivos establecen las rutas en cuanto son requeridas para transmitir la información. Finalmente los híbridos combinan los dos esquemas anteriores.

3.3.4 Enrutamiento OLSR

El esquema OLSR, del inglés Optimized Link State Routing Protocol, es un desarrollo para redes móviles ad hoc que opera de manera pro activa. Este protocolo utiliza un

mecanismo optimizado de diseminación de la información conocido como “multipoint relays”, dicha mejora difiere de la diseminación directa ya que no permite que todos los nodos retransmitan los mensajes. Contrario a eso, OLSR utiliza un grupo selecto de nodos vecinos al nodo a transmisor conocidos como MPR´s, los cuales son responsables de retransmitir la información. Este método resulta en un eficiente mecanismo para el control de flujo en la diseminación por toda la red porque reduce considerablemente la cantidad de transmisiones requeridas para mover la información.

Hay varias formas de escoger los MPR´s, pero independientemente de esto, el conjunto de MPR´s para un nodo debe verificar que son capaces de alcanzar a todos los vecinos situados a una distancia de 2 saltos desde el nodo que los calcula, esto quiere decir que los nodos seleccionados tienen una responsabilidad especial cuando declaran información sobre el estado de los enlaces. Los nodos seleccionados por algún nodo vecino como MPR´s, anuncian esta situación de forma periódica mediante mensajes de control, de esta forma un nodo avisa al resto de la red si cuenta con alcance suficiente para aquellos nodos que lo han seleccionado como MPR´s. Durante el cálculo de las rutas, los MPR´s son utilizados para formar la ruta hasta algún destino dentro de la red. Por otro lado, estos nodos seleccionados pueden ser también empleados para la eficiente diseminación de mensajes de control en la red.

De lo anterior se puede entonces decir que un nodo en una red implementada con OLSR puede comportarse en una de las dos siguientes formas; como un MPR seleccionado por su vecino más cercano (1-hop neighbor), para que retransmita todos los mensajes multidifusión que son recibidos desde el nodo que lo seleccionó, desde luego una vez que haya comprobado que no se trata de un mensaje repetido o que tenga un tiempo de vida (TTL) nulo. O bien, como un nodo selector de MPR que utiliza al mismo para diseminar mensajes.

OLSR es indicado para escenarios donde el tráfico es aleatorio y esporádico. Otra ventaja es que al ser un protocolo pro-activo cuenta con tablas actualizadas de enrutamiento en todo momento, esto se traduce en rutas disponibles inmediatamente en el momento que éstas son necesitadas. Además, comparte la desventaja de cualquier protocolo pro-activo, requiere una carga adicional en la red inalámbrica debido a la transmisión periódica de mensajes de control.

La mínima información sobre el estado de los enlaces requerida es, que todos los nodos, seleccionados como MPR, deben confirmar los enlaces con sus selectores. Toda información adicional, si es que hay, podría ser utilizada para tener redundancia. OLSR está diseñado para trabajar en un ambiente completamente distribuido y no depende de ninguna entidad central. El protocolo no requiere de transmisiones con alta confiabilidad ya que cada nodo envía mensajes de control de manera periódica, y por lo tanto puede tolerar una pérdida en proporciones razonables a la cantidad de información. Dichas pérdidas ocurren frecuentemente en comunicaciones por radio y en su mayoría son causa de colisiones o interferencias [20].

Cada nodo mantiene informado a sus vecinos acerca de sus MPR´s seleccionados, al grupo de nodos escogidos se les conoce como set de MPR´s. El resto de los nodos obtienen las actualizaciones sobre este set mediante un conjunto de mensajes "Hello" que son emitidos en forma periódica por cada nodo con MPR´s asociados. Además los mensajes Hello tienen un salto de alcance y su intercambio permite a cada nodo de la red conocer los nodos situados a 1 y 2 saltos de distancia.

OLSR utiliza además un segundo tipo de mensajes de control, éstos reciben el nombre de mensajes TC, mensajes de topología. Estos mensajes que también se envían de manera periódica y asíncrona, informan sobre cambios en la topología. Contrario a los mensajes Hello, los mensajes TC tienen alcance global y se dispersan por toda la red, de manera que el conjunto de mensajes recibidos por un nodo, le permiten reconstruir su información topológica y calcular un nuevo árbol de caminos, en algunos casos mediante el uso del

algoritmo Dijkstra. La diseminación de estos mensajes se lleva a cabo con la ayuda de los MPR´s de cada nodo [19].

3.4 Sistema de Posicionamiento Global

El GPS es un sistema de posicionamiento global que tiene la capacidad de ubicar cualquier punto sobre la superficie terrestre de manera muy precisa, existen receptores GPS con márgenes de error de algunos metros aunque también hay algunos bastante precisos que son capaces de ubicar vehículos u objetos con errores de apenas algunos milímetros.

Este sistema está conformado por 24 satélites artificiales que orbitan a 20.000 km sobre la tierra y están distribuidos en seis órbitas, o sea que se tienen 6 satélites por órbita. Dicha distribución espacial le permite enlazar a por lo menos 8 satélites desde casi cualquier punto del globo. Los satélites recorren dos órbitas al día y todo el tiempo se encuentran emitiendo ondas de radio sobre la tierra con información acerca de su posición y del momento en que fue medida. Desde la tierra, cualquier dispositivo electrónico puede usar un receptor de GPS para decodificar las señales y obtener una posición de referencia, formando una red de posicionamiento para uno o varios móviles.

El posicionamiento se interpreta con el uso de coordenadas para establecer una localización con respecto a una posición 0º, dichas coordenadas son una especie de sistema cartesiano de dos ejes, por un lado se tiene la latitud que es medida con respecto al Ecuador; por otro lado longitud que se mide con respecto al meridiano Greenwich [21].

3.5 Protocolos TCP/IP

3.5.1 Modelos OSI y TCP/IP

Un protocolo es un set de regulaciones que describen varios elementos en un sistema de comunicación, entre ellos el formato y la estructura de los paquetes de información, el método para compartir la información acerca de las rutas hacia otras redes, describen cómo y cuándo los mensajes de control (entre ellos las notificaciones de error), son difundidos entre los dispositivos; finalmente definen la inicialización y culminación de las sesiones de comunicación. Por otro lado el estándar es aquel desarrollado e implementado para asegurar que productos de diferentes manufactureros puedan trabajar juntos.

Durante un intercambio de información, diferentes protocolos pueden trabajar juntos para garantizar que los mensajes están siendo recibidos e interpretados por las partes de una red. Los protocolos pueden clasificarse según la función que desempeñan en:

- **Protocolos de aplicación**: gobiernan la manera en que el cliente y el servidor interactúan, postulando cómo deben ser las solicitudes y respuestas intercambiadas.
- **Protocolos de transporte**: existen dos principales, el TCP y el UDP. Éstos manejan las conversaciones entre un cliente y un servidor, dividen el paquete en pedazos pequeños llamados segmentos. El TCP tiene un mayor control sobre la entrega y es capaz de retransmitir en caso de error, además ordena los segmentos de forma que se entrega al destino en el mismo orden que tiene en el origen. Por otro lado, UDP construye datagramas que no llevan un orden y no tienen un nivel de confiabilidad tan alto como el de los TCP.
- **Protocolo de internet**: IP, internet protocol, toma los segmentos y los encapsula en paquetes, asignándoles una dirección de origen y destino con las cuales escoge las rutas.
- **Protocolos de red**: está conformado por dos partes, el enlace de datos y los estándares para acceso físico al medio. El enlace de datos toma los paquetes de IP

y les da un formato adecuado para transmitirlos sobre el medio. Los estándares gobiernan sobre cómo las señales son enviadas e interpretadas por los clientes [26].

Como se nota en la descripción anterior, cada protocolo se clasifica según las funciones que desempeña, además existen protocolos que si bien presentan algunas diferencias, comparten un conjunto de funciones en común. Esto forma el concepto de capa en un modelo, una capa es un nivel lógico de la comunicación que comparten varios protocolos. En la figura 3.7 se ilustran las 4 capas del modelo TCP/IP.

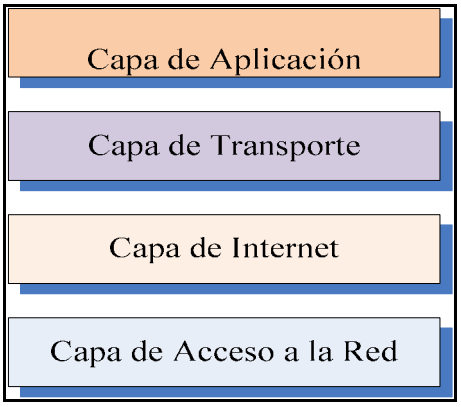


Figura 3.7 Modelo de protocolos TCP/IP.

El conocido TCP/IP es un modelo de protocolos, debido a que éste describe las funciones de cada capa en el conjunto de protocolos de TCP/IP. La capa de aplicación es aquella que contiene a los protocolos encargados de presentar los datos al usuario, la capa de transporte contiene a los protocolos que dan soporte a las comunicaciones entre varias redes, la capa de internet define las rutas para los paquetes, finalmente la capa de acceso a la red es un conjunto de protocolos y estándares para el control del hardware. La definición de los estándares es abierta, los protocolos son discutidos en un foro público conocido como RCF´s (request for comments), éstos contienen todas las especificaciones formales y de uso de los protocolos [26].

Un modelo de referencia provee una referencia en común para mantener la consistencia entre todos los protocolos de red y de servicios. El objetivo principal de un modelo de

referencia es mantener claro cuáles son las funciones y procesos involucrados. El modelo OSI, Open System Interconexion, es usado como modelo de referencia en la teoría de redes.

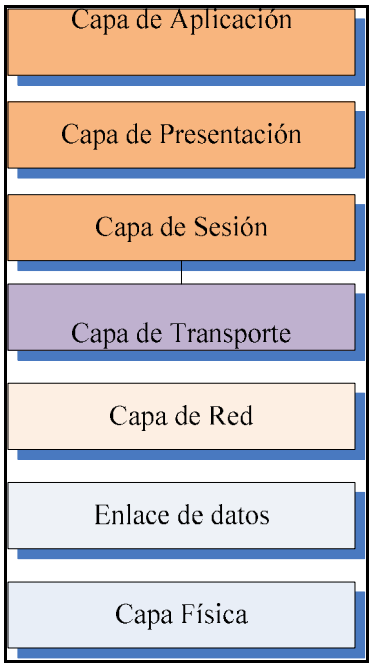


Figura 3.8 Modelo de referencia OSI.

La capa de aplicación del modelo TCP/IP es seccionada en 3 capas con funciones más específicas, la capa de sesión organiza el diálogo y el intercambio de información, en la capa de presentación se define cómo se van a representar las datos para la aplicación, finalmente la capa de aplicación se refiere a la interfaz con el usuario. La capa de transporte es aquella donde se definen los servicios de segmentación y re-ensamblaje por otro lado, la capa de red provee los servicios para el intercambio de las piezas individuales. La capa de Acceso a la red del modelo TCP/IP es vista como dos capas en el modelo OSI, por un lado la capa de enlace de datos que describe los métodos para el intercambio de datos entre los dispositivos, y por otro lado la capa física, que define las especificaciones eléctricas, funcionales y procesos para la utilización del medio físico que se utiliza como canal [27].

La figura 3.8 describe la jerarquía de las siete capas del modelo de referencia, se puede observar cómo la información debe pasar desde el medio físico hasta su destino y atravesar seis capas más para llegar hasta la aplicación utilizada por el usuario. Sin embargo no siempre son utilizadas todas las capas en una comunicación, y por lo general algunos estándares se definen para una o dos capas. Tal es el caso de Ethernet, que define los métodos para las capas de enlace de datos y física.

3.5.2 Ethernet y control de acceso al medio (MAC)

La distinción entre el dominio de colisión y el dominio broadcast viene dado porque en una simple red Ethernet se usa un sistema de transmisión compartido. Esto se refiere a que en una conexión simple (sin switches o puentes), los datos son transmitidos a todos los nodos conectados a la red, éstos posteriormente revisan la dirección de destino MAC y desechan las tramas que no correspondan a sus direcciones. Si dos nodos transmiten al mismo tiempo se produce una colisión y los repetidores propagan las tramas entre toda la red sin prevenir la colisión. Para lidiar con esta situación los switches actúan como buffers, analizan y reciben las tramas de cada segmento de red. Si una trama está destinada a un nodo en otro segmento de red entonces son enviados sólo a ese segmento de red, esto divide los dominios de colisión en dominios más pequeños, optimizando el uso del canal. Sólo los mensajes de multidifusión son transmitidos a todas las salidas de un switch. En una V-LAN los nodos de distintas subredes suelen comunicarse con mayor frecuencia y por ende a pesar de reducir el tamaño de los dominios de colisión, una V-LAN sigue siendo un solo dominio de colisión.

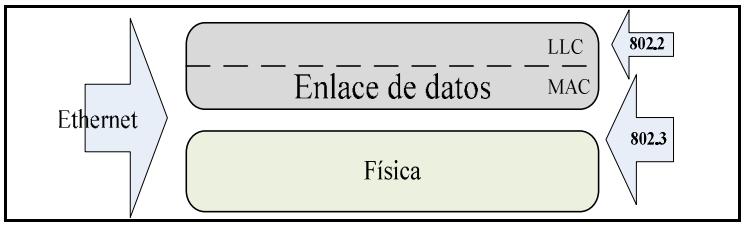


Figura 3.9 Capas involucradas en Ethernet.

Como se describe en la figura 3.9 Ethernet funciona sobre la capa física y la capa de enlace de datos. El formato y direccionamiento del esquema básico es el mismo en todas las versiones. El estándar 802.2 con algunas modificaciones conforman el 802.3, éste define las especificaciones de Ethernet. La capa de enlace de datos se divide en dos partes. La subcapa superior es el control del enlace lógico (LLC), que es conocido como el "driver" de la tarjeta de control de red y que identifica el protocolo de red. La subcapa más abajo es el control de acceso al medio (MAC), que está implementado en hardware y se encarga de delimitar las tramas, añadir la dirección MAC, agregar el chequeo por redundancia cíclica (CRC), y colocar las señales en el medio.

Ethernet utiliza un bus de multi-accesos, para descartar los paquetes que no son para un nodo en específico se utiliza la dirección MAC, el método usado por el estándar para el acceso al medio es conocido como CSMA/CD, carrier sense media access with collision detect. Significa que es un protocolo de acceso múltiple que monitorea la portadora, lo hace definiendo cada equipo para que verifique la presencia de una portadora en la línea antes de transmitir. Si se produce una colisión interrumpen su comunicación y esperan un período de tiempo aleatorio para transmitir [28].

El estándar de Ethernet ha sido modificado para trabajar con diferentes medios físicos y a distintas velocidades, hasta hace unos años los más comunes fueron 10Base-T y 100Base-TX diseñados para trabajar con cables UTP trenzados y conectores RJ-45. En general una trama de Ethernet, con algunas variantes en ciertos casos, se ve como en la figura 3.10.

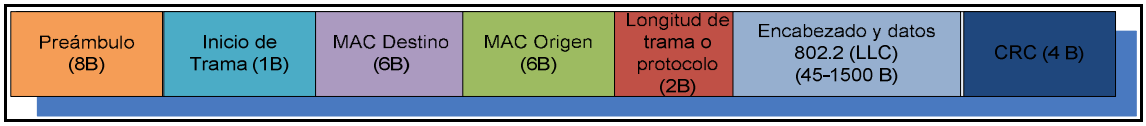


Figura 3.10 Estructura de una trama Ethernet.

3.6 BSD Sockets y el Stack TCP/IP Microchip

Un socket es una interfaz de programación para codificar la comunicación entre hosts usando el concepto de Socket de Internet, esta implementación de interfaz es la API original TCP/IP. Esta muy orientado al control y acceso de la interfaz por software (raw sockets), de los sockets. En palabras más sencillas, un socket es una forma de acceso a los servicios de red, que proporciona una interfaz de comunicación abstracta que se implementa sobre la capa de transporte, tal y como se muestra en la figura 3.11.

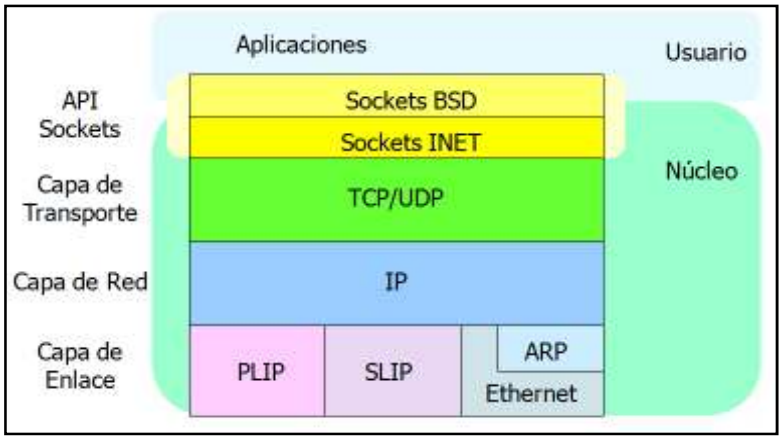


Figura 3.11 Sockets sobre TCP/IP. Tomado de [30]

3.6.1 Sockets BSD

Un socket BSD, conocido también como Berkley socket, es una interfaz de programación que comprende una librería para el desarrollo de aplicaciones en lenguaje de programación C, estas librerías ejecutan un proceso de interacción que facilita la comunicación a través de una red. Estos sockets que trabajar en diferentes implementaciones de sistemas operativos, fueron desarrollados en la Universidad Berkley de California para el uso en sistemas Unix [31].

En la capa BSD se realizan las gestiones de ficheros virtuales y administración de la estructura de los datos en general. La capa sockets INET gestiona los extremos de

comunicación para los protocolos TCP y UDP. Dependiendo del socket, la capa que hay por debajo es la capa de TCP, UDP o directamente IP. La creación de un socket conlleva a una secuencia que va de la siguiente forma:

1) Creación de la estructura socket.
2) Establecimiento del tipo de socket (SOCK_STREAM, SOCK_DGRAM)
3) Llama a la construcción de la estructura.
4) Devuelve el socket que se ha creado [30].

Los sockets Berkley pueden operar de manera bloqueadora y no bloqueadora. Una sentencia bloqueadora no regresará el control hasta que se haya recibido o enviado parte o todo el dato. Es normal que para un socket bloqueador de envío no se envíe toda la información. La aplicación debe revisar el número de bytes que se han recibido o enviado y reenviar toda la información que no se haya terminado de procesar. La operación bloqueadora tiene el inconveniente de provocar que un programa se enclave cuando se espera por información que nunca ha de llegar. Por eso es necesario tener especial cuidado cuando se utilizan este tipo de sentencias por ejemplo, para una comunicación TCP es imprescindible asegurarse que existe una comunicación, antes de ejecutar una instrucción de aceptación de mensajes.

Otra consideración importante es que el sistema no libera los recursos asociados con el socket hasta que no se ejecute un cerrado del socket, y sin embargo solo es desalojado el espacio de memoria relacionado a la interfaz del socket y no el socket en sí, esto facilita volver a usarlo bajo otra asignación, luego de un rango de 4 minutos el socket en sí es destruido por el kernel del sistema [31].

3.6.2 Descriptores y direccionamiento para sockets BSD

Un descriptor para un socket es, en términos muy generales, como una especie de encabezado, puede compararse con loa encabezados de un archivo (*FILE*handle*). Dicho descriptor es utilizado para ejecutar llamadas adicionales al socket dentro de varias instrucciones como lo son la instrucción de conexión, lectura, escritura y para cerrar el socket. La forma que tiene un descriptor para un socket BSD aparece en la figura 3.12:

```
struct addrinfo {
    int              ai_flags;     // AI_PASSIVE, AI_CANONNAME, etc.
    int              ai_family;    // AF_INET, AF_INET6, AF_UNSPEC
    int              ai_socktype;  // SOCK_STREAM, SOCK_DGRAM
    int              ai_protocol;  // use 0 for "any"
    size_t           ai_addrlen;   // size of ai_addr in bytes
    struct sockaddr *ai_addr;      // struct sockaddr_in or _in6
    char            *ai_canonname; // full canonical hostname

    struct addrinfo *ai_next;      // linked list, next node
};
```

Figura 3.12 Estructura descriptora para un socket BSD. Tomado de [32]

Como puede verse en la figura 3.13, el descriptor para un socket se presenta como una estructura de tipo entero que agrupa la información de direccionamiento del socket necesaria para especificarlo por completo. Por otro lado, la estructura "*sockaddr*" contiene la información de dirección de un socket para varios tipos de socket, tiene la forma siguiente:

```
struct sockaddr {
    unsigned short    sa_family;    // address family, AF_xxx
    char              sa_data[14];  // 14 bytes of protocol address
};
```

Figura 3.13 Estructura de dirección para un socket BSD. Tomado de [32]

De donde se puede comentar que *sa_data* contiene la dirección destino y el puerto para el socket sin embargo, es tedioso empaquetar la dirección en *sa_data* de forma manual. Para lidiar con esto se creó una estructura paralela llamada *struct sockaddr_in*; para ser usada con IPv4. Un puntero a *struct sockaddr_in* puede moldear un puntero a *struct sockaddr* y viceversa. Aun cuando la función *connect()* admite un *struct sockaddr*, es posible usar un *struct sockaddr_in* como parámetro de entrada.

```
struct sockaddr_in {
    short int          sin_family;  // Address family, AF_INET
    unsigned short int sin_port;    // Port number
    struct in_addr     sin_addr;    // Internet address
    unsigned char      sin_zero[8]; // Same size as struct sockaddr
};
```

Figura 3.14 Estructura de dirección paralela de un socket BSD. Tomado de [32]

De las figuras 3.14 y 3.13 es importante aclarar que el entero *sin_family* es equivalente a *sa_family* en la estructura *sockaddr*. El arreglo de 14 bytes se descompone en el número de puerto (*sin_port*), la dirección IP (*sin_addr*) y un arreglo de 8 bytes cuyo único fin es el de hacer coincidir los tamaños entre las dos estructuras y está lleno de valores cero. La estructura correspondiente a la dirección IP es en realidad una unión bastante particular con la siguiente forma:

```
// (IPv4 only)
// Internet address
struct in_addr {
    uint32_t s_addr; // entero de 32-bit (4 bytes)
};
```

Figura 3.15 Estructura unión para la dirección IP de un socket. Tomado de [32]

De manera que si por ejemplo se ha decidido declarar a "ina" como de tipo estructura sockaddr_in, entonces ina.sin_addr.s_addr se refiere a la dirección IP de 4 bytes. Es importante notar que aun cuando el sistema utilice la complicada unión, es posible hacer referencia a la dirección IP de 4 bytes en el mismo modo que se ha hecho para el ejemplo de "ina", lo cual resulta muy cómodo luego de haber usado unas cuantas definiciones (#define) [32].

3.6.3 Microchip TCP/IP Stack con BSD Socket API

El stack de Microchip con BSD proporciona las librerías para los sockets de comunicación TCP/IP. La idea de Microchip es que la se cuente con una interfaz común de programación de forma que las aplicaciones sean portables y soportadas por completo en varias plataformas, por ejemplo que una aplicación de red escrita para un ambiente PC también pueda ser compilada en una ambiente de embebido.

API- socket: crea el socket de comunicación, esta función retorna un descriptor para el socket creado. Recibe como parámetros principales el tipo de dirección (actualmente solo es posible utilizar AFI_NET), el tipo de socket (puede ser un socket para datagrama o para

stream), y el protocolo (para stream sockets debe ser TCP y para datagram sockets debe ser UDP)

API- bind: asigna una estructura de dirección y nombre a un socket no asignado. Retorna un valor de cero si la operación es exitosa y un número negativo en caso contrario. Recibe como descripción el descriptor del socket a nombrar, un puntero a la estructura que contiene la dirección local del socket y el tamaño de dicha estructura.

API- listen: especifica un sockets para ponerlo en modo de escucha, esta función es usada para TCP. Recibe como parámetros al descriptor del socket, una cantidad máxima de conexiones a solicitar, y retorna un cero si se la operación es exitosa o un número negativo en caso contrario.

API- accept: acepta las conexiones colocadas en la cola de solicitudes para un socket en estado de escucha. Recibe el descriptor, el nombre de la estructura con el direccionamiento del socket, el tamaño de esa estructura y retorna un valor no negativo si la función se ejecuta de forma correcta.

API- recvfrom: procesa la información pendiente que ha sido colocada en cola para un socket, normalmente es usada para recibir mensajes sobre datagramas sin embargo, puede ser utilizada para comunicaciones TCP. De no resultar nula, la función retorna la dirección del origen del datagrama colocándola en la estructura de direccionamiento. También se modifica el tamaño de esa estructura de direccionamiento. Si no existen datos disponibles en el socket, la función retorna cero. Si la función se ejecuta sin problema, el número de bytes copiados al buffer de aplicación es retornado.

API- sendto: es usada para enviar datos a través de un socket de datagramas, la dirección de destino está especificada dentro de una estructura de direccionamiento señalada por los parámetros de entrada “to” y “tolen”. Donde el primero apunta a la estructura de dirección destino y el segundo especifica el tamaño de dicha estructura. Además recibe como parámetros el descriptor del socket de envío, el buffer desde donde se leerán los bytes a enviar. Podría retornar error en caso de no haber suficiente memoria para alojar el buffer del paquete o un error general.

API- connect: asigna la dirección del otro punto para la comunicación. Para stream sockets la conexión es establecida entre ambos puntos. Para datagramas un filtro de dirección es establecido entre los dos puntos hasta que sean cambiadas las opciones con otra función connect.

API- send: similar a "send to" pero para usarse únicamente en stream sockets, requiere de menor cantidad de parámetros de entrada ya que asume que la conexión entre los hosts está preestablecida.

API-recv: como en el caso anterior, es la versión de la función que sólo puede usarse para stream, de igual forma retorna la cantidad de bytes copiados al buffer de aplicación en caso de que no hayan errores.

API- closesocket: cierra un socket específico, cualquier información almacenada con respecto a este socket es eliminada, inclusive los datos en los buffers de envío o recepción.

API-InitStackMgr: realiza las inicializaciones de todos los módulos que conforman el stack, esta aplicación debe ser llamada antes de ejecutar cualquier otra.

3.7 Descripción del sistema anticolisiones Acumine

En esta última sección del capítulo se realiza un análisis del sistema anticolisiones Acumine, el principal elemento de interés en este análisis es el sistema de comunicación que utiliza la tecnología Acumine, ya que el desarrollo de este proyecto se desenvuelve en la capa de comunicaciones del sistema anticolisiones. Por ende, serán mencionados muchos de los conceptos vistos en las secciones anteriores. Una vez entendido por completo el sistema de comunicación en cuanto a su funcionamiento, topología y protocolos, es posible aclarar cuál es el papel que jugará el nodo enrutador dentro de las comunicaciones del sistema anticolisiones Acumine. La figura 3.16 muestra un diagrama general de los bloques funcionales del sistema Acumine.

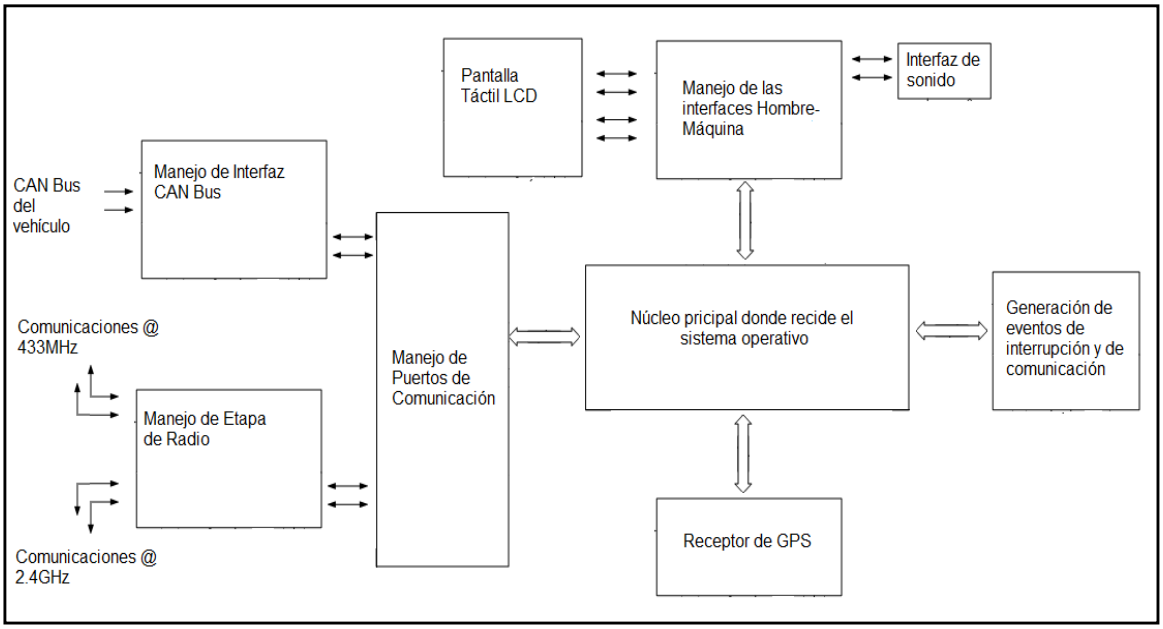


Figura 3.16 Diagrama de bloques de las cajas Acumine.

El bloque principal de la figura 3.16 es el bloque sobre el cual corre un sistema operativo que ejecuta muchas de las tareas con ayuda de periféricos y controladores externos que le colaboran con gran parte del trabajo sin embargo, a pesar de tener unidades externas que manejan algunos de los eventos e interfaces, sería demasiado trabajo lógico para un hardware que basara su funcionamiento en máquinas de estado paralelas. Es por ello que los investigadores del IIIE decidieron utilizar una plataforma que soporte algún sistema de libre distribución. Dicho hardware debería incorporar algunas de las unidades para la radiocomunicación de 2.4GHz. Como se ha mencionado anteriormente, el sistema Acumine trabaja con una red de comunicaciones mesh, entonces podría utilizarse el hardware de un router linksys como punto de acceso. Los modelos de las series WRT54G (véase el anexo 1), no fueron diseñados originalmente para usarse en una zona abierta ni con redes mesh, sin embargo es muy utilizado por su bajo costo y fácil manejo. Existen muchas distribuciones del firmware para los WRT, entre ellas el FreifunkFirmware que ya viene directamente con soporte para mesh. Además es un dispositivo que se puede comprar con menos de 100 dólares y cuyo firmware es sustituible por versiones de desarrolladores independientes. Entonces una tarjeta del WRT con algunas adaptaciones de hardware se emplea como punto de acceso en la red mesh de Acumine [22].

Es de interés aclarar que la red mesh formada por los nodos de la tecnología Acumine utiliza el protocolo de enrutamiento OLSR, por ello se le dio especial énfasis en secciones anteriores de este capítulo. Freifunk es un desarrollo de libre distribución que trabaja con este protocolo, sumando esto al hecho de que parte del hardware utilizado es el WRT de linksys, y que éste es compatible con sistemas operativos derivados de GNU/Linux; se tiene por justificado la escogencia de este hardware por parte de la gente de Acumine.

Como se ha mencionado anteriormente, el sistema Acumine tiene como objetivo principal evitar colisiones entre vehículos de gran tonelaje dentro de un ambiente de trabajo rodeado de riesgos y condiciones adversas. Para ello esta tecnología necesita llevar a cabo la localización de todas las unidades que se encuentren en el campo de trabajo, tanto las que se encuentran en movimiento como las que se encuentren estáticas, dicha localización es realizada utilizando un sistema de GPS en cada una de las unidades llamadas cajas y en cada nodo liviano. Con las posiciones obtenidas mediante un receptor y decodificador GPS, como se ve en uno de los bloques de la figura 3.16, la unidad anticolisiones puede estimar la ubicación del camión en el cual se encuentra conectada.

Pero también es indispensable que un sistema anticolisiones conozca las posiciones de las otras unidades vehiculares en el campo de trabajo, por ende es necesario establecer un sistema de comunicación por donde la información sobre las localizaciones de los otros pueda ser diseminada por toda una red. Es necesario considerar que la red debe ser de carácter dinámico ya que, al haber movimiento de muchos nodos de la red, se presentan variantes en la topología de la red. Esto significa que rutas fijas que no sean capaces de reconstruirse o adaptarse serían inútiles y se produciría una inminente pérdida de la información. Debido a la velocidad tan baja a la cual operan los camiones y otros móviles, la rapidez con la que se actualicen los datos no es tan crítica como la veracidad de dichos datos, ya que una falsa información podría resultar en los accidentes que precisamente se tratan de anular. Es por esta razón que el sistema anticolisiones Acumine, cuenta con una red de respaldo que le permite a las unidades comparar la información y desmentir falsas ubicaciones. Además la red de respaldo que emite mensajes a sus vecinos más cercanos le

permite al protocolo de enrutamiento verificar algunas de las rutas trazadas mediante los algoritmos propios del OLSR.

Como última función asignada a la banda de 433MHz está la de transportar mensajes de alerta hacia los vehículos que se encuentran en los alrededores. Estas importantes funciones justifican el aumento en el consumo de ancho de banda demandado para el funcionamiento del sistema Acumine, ya que al usar la frecuencia de 2.4GHz para implementar una red mesh y otra a 433MHz para respaldarla, se consume mucho más ancho de banda que utilizando una sola. De manera que cualquier funcionalidad extra que se le pueda dar a la banda auxiliar, sin sobre cargar las comunicaciones por este canal claro está, es bienvenida para los investigadores del IIIE. Es allí donde reside la justificación de este proyecto; utilizar los mensajes con la posición y los mensajes de alerta para re-direccionarlos hacia una estación de control cuando un vehículo se acerca a la misma, le agrega dos funcionalidades más al sistema, justificando aún mejor el consumo de ancho de banda. De forma que hasta este punto es posible describir el funcionamiento del sistema Acumine que fue cuna para la elaboración de este proyecto de graduación.

Para Acumine, si bien la red OSLR construye todas las tablas de enrutamiento utilizando las funciones de los MPR´s que son propias del protocolo, también se aprovechan los mensajes ping emitidos a 433MHz para hacer una comparación entre las rutas lógicas que trazan los algoritmos de cálculo incorporados por el protocolo, con las rutas basadas en la ubicación espacial (como una herencia de los protocolos Location Based), de manera que se combinan en cierta forma dos métodos para obtener una optimización de la red. Sin embargo debido a las políticas de confidencialidad de la marca Aucmine, no se presentan detalles de este procedimiento de optimización en este documento.

Capítulo 4: Procedimiento Metodológico

4.1 Reconocimiento y definición del problema

Para comenzar con el planteamiento del problema se utilizó información adquirida antes de realizar el viaje hasta el lugar de trabajo, dicha información fue adquirida por medio de correo electrónico. La cooperación del Profesor Dr. Pablo Sergio Mandolesi, que junto al Dr. Fabio Manson llevan a cabo la dirección de los proyectos, fue fundamental ya que el Dr. Mandolesi proporcionó información previa que permitió hacerse una idea sobre el entorno donde del proyecto. Sin embargo, al tratarse de un proyecto en otro país, la distancia siempre fue una limitante ya que imposibilitó llevar a cabo mediciones u observaciones en el entorno de trabajo.

El momento en que realmente se consolidó una definición sólida del problema, fue tras estar en contacto con el equipo de investigación durante varias semanas y luego de obtener un entendimiento del funcionamiento del sistema Acumine. Otro punto que sería provechoso mencionar en este apartado es, que si bien se realizó el viaje hasta la UNS, sede dónde trabaja el equipo de investigación de Acumine, el verdadero entorno de trabajo para la aplicación del sistema es en Chile, esto significa que en general todo planteamiento basado en observaciones de campo, tuvo que llevarse a cabo con base en las experiencias otros miembros que llevan años trabajando en el proyecto y que por ende, han hecho múltiples visitas a la mina ubicada en Chile.

4.2 Obtención y Análisis de la información

La metodología aplicada para llevar a cabo la investigación inicial del proyecto básicamente se basó en la recopilación de información técnica mediante entrevistas con los miembros del equipo de investigación. Al tratarse de un trabajo desarrollado por un equipo de investigación, es necesario considerar que las ideas van cambiando y que existen

múltiples diseños, algunos en proceso y otros ya propuestos. Por ello es necesario que algunos detalles en el diseño vayan cambiando.

La recopilación de la información detallada comenzó con la lectura de algunas tesis de grado realizadas por parte de otros estudiantes, y que posteriormente se postularon como etapas del proyecto Acumine, algunos de esos estudiantes ahora son parte del equipo, lo que significa que se contó con la oportunidad de realizar consultas sobre sus trabajos a lo largo de toda la fase inicial de la investigación. Las reuniones al principio fueron incentivadas por el profesor tutor y se realizaban en forma grupal, de manera que todos los miembros del equipo pudieran acotar sus ideas, estas reuniones tuvieron como objetivo que el autor se involucrara con el entorno del proyecto y con el estado actual del sistema. Además las primeras reuniones permitieron socializar con los miembros del equipo, esto es importante ya que siempre hay que establecer relaciones humanas para luego tener más facilidad y acceso a la información por parte de los investigadores.

Posteriormente las reuniones comenzaron a ser más individuales ya que conforme fue pasando el tiempo, se hacía más fácil identificar quién era el más adecuado para cada tipo de consulta. A parte, cada investigador era capaz de proporcionar buenas referencias bibliográficas de donde se extrajo gran parte de la información inicial. Las fuentes principales de dicha información por lo general consistieron en páginas de internet, hojas de datos, notas de aplicación y erratas.

4.3 Evaluación de las alternativas y síntesis de una solución

Para evaluar las alternativas de diseño se planteó un primer bosquejo del diseño, donde se consideraron opciones tanto para el hardware como para el software. Para llevar a cabo este planteamiento muy generalizado se tomaron algunas sugerencias del profesor tutor, sugerencias que el mismo realizó basándose en su experiencia y en la herramientas de hardware y de software disponibles.

Para realizar la evaluación del hardware a utilizar se utilizaron las hojas de datos de los fabricantes y las notas de aplicación. Una vez que se decidió el hardware a utilizar se procedió a realizar propuestas para el funcionamiento del sistema, esto va muy ligado al software del dispositivo. En este punto fue importante evaluar la disponibilidad de software de programación, simulación y debug. Finalmente la escogencia de un hardware ayudó en alguna forma a delimitar el número de opciones de software.

Otro factor muy importante para la evaluación de alternativas fue el factor tiempo. En cada solución que se propondría era necesario considerar el tiempo que se tardaría en comprar o fabricar las herramientas de hardware necesarias. El factor tiempo jugó un papel clave a la hora de determinar si era más viable adquirir algún hardware o realizarlo en los laboratorios del DIEC. Tras sintetizar una solución fue necesario descartar opciones de hardware, básicamente por problemas con los tiempos de entrega ofrecidos por los fabricantes.

Los instrumentos de medición también juegan un papel importante en la síntesis de una solución, ya que era estrictamente necesario realizar pruebas con forme el dispositivo fuera evolucionando, fue entonces cuando se hizo muy importante la familiarización con el equipo. Por ejemplo para realizar mediciones en radio frecuencia se tuvo que trabajar con el analizador de espectros Agilent E4433B, y para realizar mediciones a través de algunas interfaces de comunicación en el hardware se usó el analizador digital MSO7104A.

4.4 Implementación de la solución

Lo primero que se decidió fue adquirir hardware comercial de diseño mientras se diseñaba un hardware independiente, esto para evitar retrasos en construcciones de hardware se y facilitar una forma de realizar las pruebas del software a utilizar y el software a diseñar, descartando así errores en el hardware. Sin embargo mientras se

desarrollaban dichas pruebas se fue trabajando en paralelo con el diseño de un prototipo y la obtención de todos los componentes involucrados.

Inicialmente las pruebas de software tuvieron como único fin familiarizarse con el embebido y el stack TCP/IP de Microchip, el cual como base para las comunicaciones Ethernet. Para la realización de dichas pruebas fue necesario el uso de herramientas de software, herramientas que funcionaban para emular terminales UDP o TCP en la PC. Por suerte éstas venían incorporadas como parte del mismo software sin embargo, dichas herramientas eran para hacer pruebas sencillas y con forme el diseño avanzó, fue indispensable realizar modificaciones sobre las mismas. Para realizar estas modificaciones bastó con averiguar el compilador con las que fueron hechas (el fabricante adjuntó el código fuente), posterior a eso se tuvieron las primeras experiencias con el C++ Builder 6, compilador con ambiente de desarrollo que permite programar, depurar y correr aplicaciones en C o C++.

A pesar de que, se contaba con cierta experiencia en el lenguaje C, fue necesario tomar algunas clases en el Departamento de Ingeniería en Informática sobre manejo de archivos y sockets. Posterior a eso fue posible correr algunas pruebas en la PC enlazada con el PICkit vía Ethernet, estas experiencias orientaron adecuadamente el diseño de software para pruebas más complejas indispensables durante las últimas fases del proyecto. Relacionado a estas pruebas del stack, también cabe mencionar que para el análisis del stack TCP/IP de Microchip con BSD fueron necesarias horas y horas de lectura de código, se realizaron diagramas y esquemas tratando de asociar la monumental cantidad de librerías, headers y módulos. Al final lo más conveniente fue implementar una variante de los diagramas de pez, dónde se describe la relación entre archivos de configuración, los encabezados y los archivos fuente. Lo anterior debido a que sólo se cuenta con un par de manuales que hablan al respecto, y lamentablemente no detalla tanto como se quisiera.

Regresando al hardware, con forme fueron pasando las semanas y el diseño de una placa con todo el hardware integrado se retrasó, se toma la decisión de fabricar un módulo de expansión para el hardware en ese momento disponible. La realización de un PCB involucró la utilización de programas como el Altium Designer, con él se llevó a cabe la edición de footprints, esquemáticos y finalmente el PCB. La edición de algunos footprints fue imprescindible debido a que no todos los componentes necesarios se encontraban disponibles en las librerías del software, y la información técnica acerca de las dimensiones y otras consideraciones se extrajo de las hojas de datos de los componentes. Además se utilizaron notas de aplicación para la selección de los componentes pasivos externos al transductor de radiofrecuencia, necesarios para el funcionamiento del transductor y el acople de la antena.

Para realizar el driver del transductor para compatible con el microcontrolador, se realizó lectura de la hoja de datos del CC1101, así como de todas las notas de aplicación y erratas sugeridas por los otros miembros del equipo que tuvieron experiencias previas con el transductor. Conforme se fue haciendo la lectura se anotaron todas las consideraciones importantes y se hicieron algunos bosquejos de diagramas de flujo para la programación en el PIC. Se hizo especial énfasis en la máquina de estados del CC1101 y en las configuraciones a los registros internos, con el fin de garantizar la compatibilidad de la comunicación con los nodos Acumine.

En cuanto a la programación, el método de la modularización fue indispensable. La separación de las rutinas no solo facilitó la lectura futura del código, sino que permitió realizar un depurado de manera más ordenada y sencilla. Para facilitar las cosas se decidió trabajar todo el sistema que debía ser como dos subsistemas omnidireccional. De manera que al juntarlos se tendría la comunicación en ambas direcciones sin embargo, al haberse trabajado inicialmente como un canal en una dirección y otro en otra, se ahorró mucho tiempo y dolores de cabeza. De igual forma, una vez que la comunicación bidireccional fue posible y se había probado repetidas veces, se procedió a incorporar las otras funcionalidades al nodo Ethmesh.

En algún punto intermedio fue necesario programar herramientas de software en la PC que permitieron confirmar la operación de algunas funciones del Ethmesh, como el cálculo del CRC. También de crearon funciones dentro del software de prueba que permitieron acceder a registros del nodo, con esto se pudo obtener información remota que revelaba información valiosa para la depuración del sistema, como por ejemplo el estatus del transductor RF, al estado de las memorias FIFO y realizar resets y vaciados de las memorias del transductor. En otras palabras, en algún punto de la implementación del diseño fue necesario crear herramientas propias de depuración y control sobre el comportamiento del hardware.

Capítulo 5: Descripción detallada de la solución

5.1 Evaluación de las posibles soluciones y solución final

5.1.1 Razonamiento para la solución final

En el primer capítulo de esta documentación se habló de una ineficiencia en el uso de la banda de 433MHz, frecuencia por la cual se despliegan mensajes entre los vehículos de la mina con información muy valiosa para el control de los vehículos y la evaluación de situaciones de alerta. El planteamiento de la solución consiste en colocar un dispositivo en las cercanías del lugar donde se realiza la descarga de materiales, de forma que cuando algún vehículo se acerque lo suficiente, el mensaje ping o el mensaje de alerta que éste emite cada cierto tiempo es capturado por dicho aparato y retransmitido hacia la estación central a través de un cable de red, una vez que el paquete llega a la estación, se distribuye por medio de un switch hacia la computadora correspondiente. Este direccionamiento es posible porque el dispositivo utilizará algún protocolo que sea soportado por la interfaz de Ethernet. En la figura 5.1 se presenta un esquema del funcionamiento.

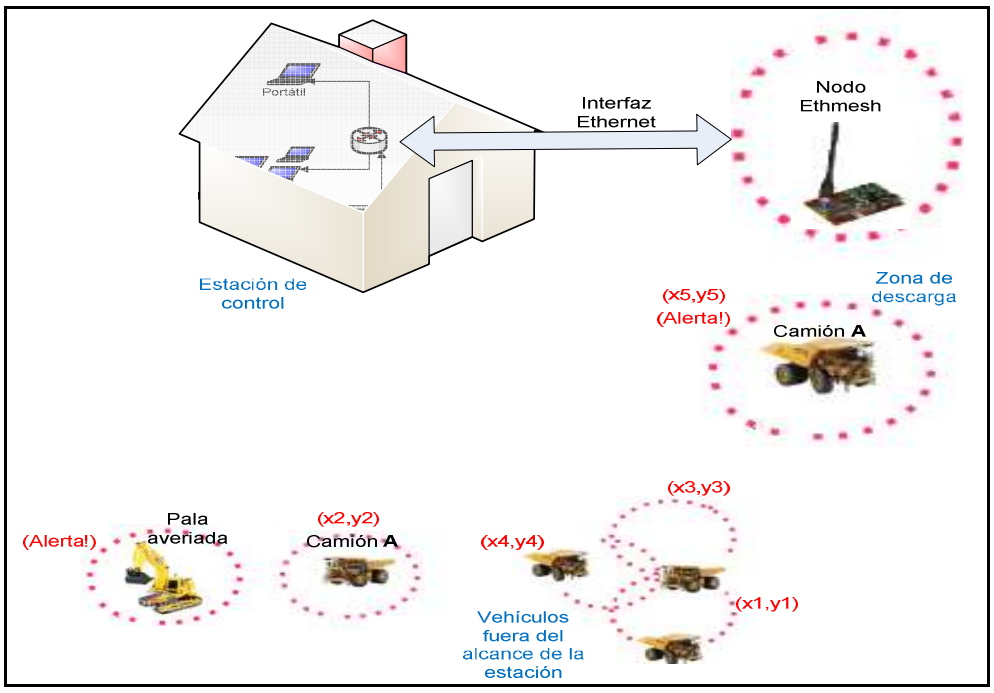


Figura 5.1 Esquema de funcionamiento del sistema de control

De la figura 5.1 pueden extraerse varias ideas importantes. Primero, el dispositivo de enlace será colocado a una cierta distancia de la estación de control, la suficiente para llegar a una zona de descarga (entre 15 y 20mts), esa es entre otras consideraciones, que se mencionarán más adelante, la razón por la que se ha escogido utilizar Ethernet.

Figura 5.2 Carga de material para transportar

Por otro lado, como se ilustra en la figura 5.2, todo camión debe acercarse a una zona de carga donde se encuentran las palas mecánicas, entonces la idea es que si se presentara alguna falla, la pala emite un mensaje que es captado por alguno de los camiones que se acercan a la zona. Dicho mensaje llevará un encabezado con un espacio asignado para determinar de qué tipo se trata (es necesario recordar que por la banda de 433MHz también se transmiten mensajes de ping). La caja instalada en el camión se dará cuenta cuando se trata de una alerta, entonces almacenará el mensaje y lo retransmitirá cuando se aproxime a la zona de descarga, la zona de descarga estará dentro del alcance del nodo de enlace y por ende el aviso podrá ser capturado y reenviado a una computadora en la estación de control asignada a estar pendiente por este tipo de mensajes de alerta. En la figura 5.1 las líneas rosas punteadas simbolizan el alcance de las transmisiones por la banda de baja frecuencia, entonces puede verse que para una pala mecánica situada lejos de la estación es imposible transmitir de manera directa a la estación central. Cualquiera podría sugerir que es mucho más sencillo utilizar la red mesh sobre los 2.4GHz, sin embargo sería poco eficiente colocar toda una jerarquía de encabezados, como los que conlleva un protocolo OLSR, a un simple mensaje de alerta. Esto significaría sobrecarga innecesaria sobre la banda de 2.4 GHz. La lógica de la solución es por así decirlo, dejar que el mensaje sea transportado de

manera mecánica y aprovechar el sistema de posicionamiento instalado en cada caja para dejarle saber el momento en que tiene que "depositar" el mensaje de alerta.

Con los mensajes de ping el razonamiento de la solución es un poco diferente, aunque también está basado en la lógica común. Si lo que se desea es llevar un registro del momento en que cada camión pasa cerca de un determinado punto, basta con colocar el dispositivo de enlace cerca de la estación de control, de esta forma desde que el vehículo se acerca a una cierta distancia se comienzan a recibir sus mensajes ping, de forma que no sólo se puede saber el momento en que llega a hacer una descarga, sino que se puede establecer un enlace de comunicación temporal (el tiempo que tarda el camión en completar la operación no es para nada despreciable), sobre la banda de 433MHz para intercambiar mensajes ping generando así un registro de rutas.

5.1.2 Visualización de la solución por etapas

Para la implementación del nodo de enlace, apodado como Ethmesh, es necesario realizar el diseño de tres etapas fundamentales, la distinción de cada etapa se hace según las funciones de cada bloque y el hardware que éste involucre. A fin de proporcionar una mejor explicación de la solución, las tres etapas básicas se ilustran en la figura 5.3.

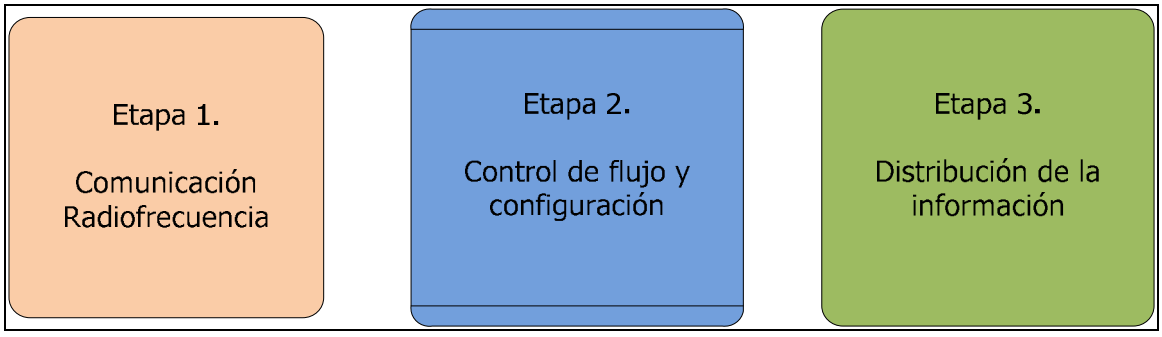


Figura 5.3 Etapas de la solución.

En la primera etapa se soluciona todo lo relacionado a la comunicación con los nodos de la tecnología Acumine a través de la banda de 433MHz. En ésta etapa se realizaron las consideraciones necesarias para decidir sobre el hardware a utilizar y el software necesario que se tuvo que programar en la etapa 2 para su manejo y por último, la estructura de los paquetes, la detección de errores, la sincronización y el preámbulo necesarios para tener compatibilidad con las cajas Acumine.

La segunda etapa contempla todo el software necesario para el manejo de ambas interfaces de comunicación, Ethernet y RF. Además conlleva a todo un diseño del control de flujo de los paquetes que implica la atención de interrupciones, revisión de los puertos de comunicación, atención a solicitudes de configuración y manejo de memoria para almacenamiento temporal de datos.

La tercera etapa reside en el mismo hardware que la segunda, se llevó a cabo utilizando TCP/IP y funciones UDP de capa de transporte, el manejo de la Ethernet fue asistido por un stack de libre distribución facilitado por el fabricante del hardware (Microchip). El hardware adquirido para la implementación de ambas etapas contiene todas las interfaces necesarias. Sin embargo, se diseña un hardware alternativo que propone una fusión de todas las etapas en una sola placa PCB, pero que por retrasos en el tiempo de entrega de algunos componentes requeridos no fue posible terminar.

5.1.3 Posibilidades para la etapa 3

La tercera etapa tiene como función primordial realizar la distribución de los datos hasta los usuarios finales. Considerando que en la estación de control se cuenta con varios computadores, y que éstos se encuentran comunicados entre sí por medio de una interfaz Ethernet, se deduce que la mejor opción para hacer llegar la información a cada una de las PC´s es aprovechando la red LAN que actualmente se tiene montada en la estación. Cualquier otro tipo de interfaz, como lo hubiera sido una conexión por puerto serie,

hubiese representado un gasto adicional de hardware y una menor escalabilidad. El concepto se describe en la siguiente figura.

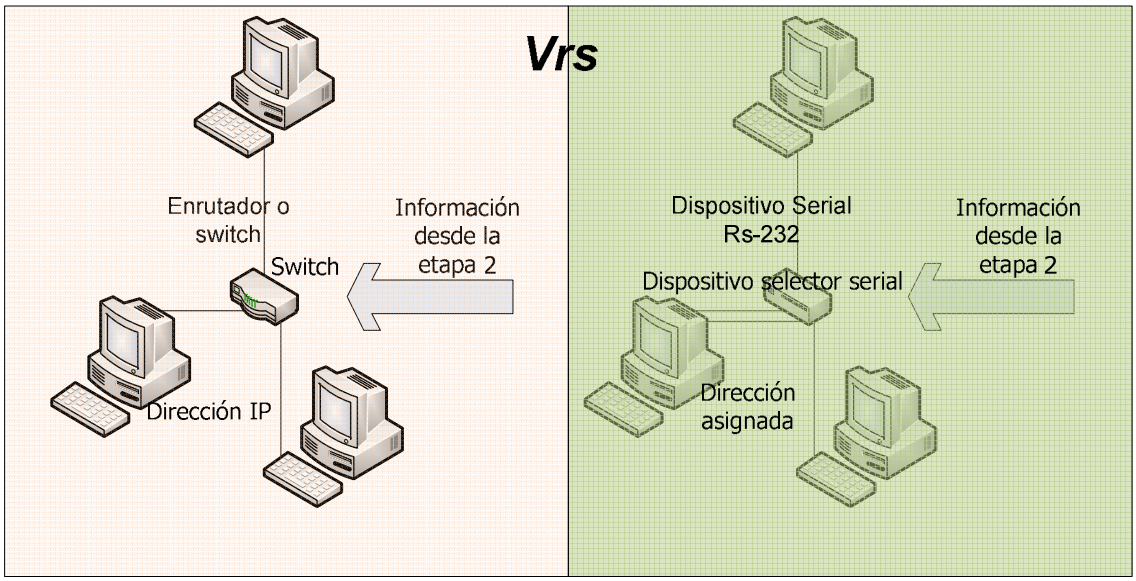


Figura 5.4 Concepto de las dos soluciones para la etapa tres.

Ambos conceptos de la figura anterior fueron considerados como opción para hacer llegar los datos a cada una de las computadoras de los usuarios. El concepto de la derecha consiste en la conexión de todas las computadoras por medio de una interfaz RS-232 en alguno de los puertos COM sin embargo, para poder llevar a cabo la entrega de los paquetes a cualquier computadora hubiera sido necesario un dispositivo especial. Dicho aparato sería un hardware dedicado repartir los paquetes a cada computador y hacerlos llegar por una interfaz RS-232, además de tener disponibles muchos puertos seriales para poder conectar las computadoras, debería tener una unidad de procesamiento para las direcciones. Esto sería una complicación innecesaria, otra de las desventajas de esta idea es que la comunicación serial por 232 no tiene una velocidad de transferencia tan rápida como la de Ethernet [34].

Otras opciones contempladas para la implementación de la tercera etapa fueron de carácter inalámbrico, por ejemplo se contempló la posibilidad de establecer un enlace por

bluetooth, el problema de usar enlaces de ese tipo es el alto costo de los equipos requeridos.

Por todo lo anterior se decide utilizar la interfaz Ethernet para realizar la entrega de los paquetes a cada una de las computadoras, esto presenta las siguientes ventajas:

- Estándar definido que incluye el manejo de las direcciones.

- La resolución de direcciones para la entrega final lo realiza un switch o router, basta con que el dispositivo de enlace coloque la dirección IP adecuada antes de enviar el paquete por Ethernet.

- El dispositivo intermedio para conectar a todas las computadoras no debe ser construido ya que se puede utilizar cualquier enrutador o switch, lo cual se traduce en un ahorro de hardware.

5.1.4 TCP vs UDP

Una vez definido que se utilizará Ethernet como interfaz física para la entrega de los paquetes a los usuarios finales, se debe ir unas capas más arriba y decidir cuál protocolo de capa de transporte utilizar. El protocolo de control de transmisión (TCP), permite realizar la entrega de segmentos de manera confiable y con control de flujo. Sin embargo consume 20 bytes de encabezado y usualmente es utilizado en aplicaciones donde cada segmento de la información total es imprescindible [35]. Por otro lado el protocolo de datagramas de usuario (UDP), no está orientado a conexión y por ende, consume menos recursos en la red ya que no hay control de flujo y tiene menor cantidad de bytes en el encabezado (8 bytes). Este protocolo es más común en aplicaciones donde la información puede llegar desordenada y no es crítico que llegue en el mismo orden en que fue segmentada [36].

En este punto sería elocuente hacer un pequeño análisis, para ello es necesario mencionar que el tamaño de los paquetes que se pasan entre los vehículos en la frecuencia de 433MHz es de máximo 256 bytes. Como se mencionó en el capítulo 3, los paquetes emitidos por la banda de baja frecuencia contienen información de posición actual, identificación o información de alerta y dispositivo en falla. Esto quiere decir que no son cantidades masivas de datos, sino más bien son mensajes cortos pero con importante información para el control de procedimientos en la mina. Por ende, se deduce que la información correspondiente a un solo paquete cabe perfectamente en un segmento TCP o UDP, de forma que los mensajes pueden ser entregados en datagramas y el orden en que éstos lleguen no afectará la validez de la información ya que es muy probable que en la mayoría de los casos un paquete de 256 bytes no sea segmentado en varios datagramas. Así se recibirán los datagramas que contienen los mensajes de varios camiones sin sobrecargar la red LAN que se encuentra en la estación de control.

5.1.5 Posibilidades en el enrutamiento de los paquetes RF hacia los computadores

El algoritmo para llevar a cabo la construcción de las rutas entre la red Ethernet y los nodos de la radio se basó en un mapeo de direcciones, la idea es que cada computadora envía una solicitud especial al nodo Ethmesh, cuando el nodo recibe esta solicitud entonces copia la dirección IP y la asocia con una dirección de nodo para la red de 433MHz. Este mapeo se almacena en una tabla de direcciones dentro del Ethmesh y es utilizado para regresar los mensajes de respuesta que vengan de un nodo inalámbrico hacia una PC específica en la red Ethernet. Dentro de cada paquete de comunicación RF se tiene una dirección destino, de forma que el Ethmesh lee esa dirección y la compara con las direcciones de nodo que tiene en su tabla. En caso de encontrar alguna coincidencia entonces toma la carga efectiva del paquete y lo retransmite por la interfaz cableada hacia la IP que estuviera asociada al número de nodo destino. Esta técnica permite a las computadoras emular el comportamiento de un nodo más dentro del canal de 433MHz, realizando la entrega y recepción de los paquetes de un lado al otro de la red cableada y la inalámbrica.

La posibilidad de emular el comportamiento de mensajes de multidifusión es fundamental para garantizar que el Ethmesh le permite a la red LAN incorporarse dentro de la banda de baja frecuencia. Esto quiere decir que si un nodo transmite con una dirección de destino 0xFF (en hexadecimal), el Ethmesh debe capturar ese paquete y reenviarlo a todos los computadores de la red LAN. Para ello podría utilizarse una IP destino corresponda a la dirección de broadcast de la red (algo como 192.168.2.255), sin embargo esta forma de hacerlo provocaría que el enrutador o switch le envíe esos mensajes a absolutamente todos los computadores sobre la LAN, pero a lo mejor no todas las PC´s están siendo utilizadas para realizar la captura de mensajes RF. Por ello es una mejor opción hacer que el Ethmesh retransmita los mensajes de multidifusión provenientes de la red inalámbrica, hacia las computadoras que se encuentran registradas en su tabla de direccionamiento, así se asegura que quienes reciben los mensajes por UDP sean las máquinas en las cuales se está corriendo la aplicación de comunicación con el Ethmesh.

5.1.6 Posibilidades para la atención de solicitudes UDP de los usuarios

Como se menciona en la sección de enfoque de la solución, el nodo Ethmesh tendrá la capacidad de ser configurado desde cualquier computador conectado a la red Ethernet dentro de la estación de supervisión. Lo anterior quiere decir que las tareas solicitadas por el usuario en la PC deben ser clasificadas entre tareas de comunicación y tareas de configuración. Entonces el nodo de enlace debería distinguir el tipo de tarea de alguna forma eficiente y así al capturar un datagrama identifica si el servicio que debe prestar es de enlace o de configuración. Para lograr esta distinción entre ambos tipos de servicio se consideraron las dos siguientes opciones. En la primera opción, el primer paquete que es enviado desde el computador, lleva dentro de sí un mensaje que indica el servicio a ejecutar. Aunque la idea parece aceptable al principio, si se piensa con detenimiento puede analizarse que para llevar a cabo la distinción de esta manera, es necesario que el mensaje llegue hasta la capa de aplicación y sea interpretado por el software que reside en el nodo Ethmesh para descartar los otros servicios. Esto representa un procesamiento e interpretación previa del mensaje, para luego llevar a cabo la decisión de cuál tarea debe entrar en ejecución, o sea un retardo en la respuesta.

En una segunda opción, este retardo operacional es eliminado aplicando la siguiente lógica. Se considera asociar el tipo de servicio a un número de puerto específico, esto quiere decir que el nodo Ethmesh utilizará tres puertos de uso libre para trabajar [37], para este diseño se han escogido los siguientes números:

- **6651**: Puerto de configuraciones, por medio de este puerto se cargan toda clase de configuración para la etapa de radio del Ethmesh.

- **6652**: Puerto mudo, llamado así pues solo se utiliza para enviar hacia una PC mientras el Ethmesh realiza un "sniffing" del canal de 433MHz, la computadora es por así decirlo muda en este puerto.

- **6653**: Puerto de Inscripciones y transmisiones, por este puerto se recibe la solicitud inicial que un usuario envía para comenzar a capturar los datos, además se reciben los mensajes para ser dirigidos hacia un nodo específico.

En la figura siguiente se ilustra el concepto aquí descrito.

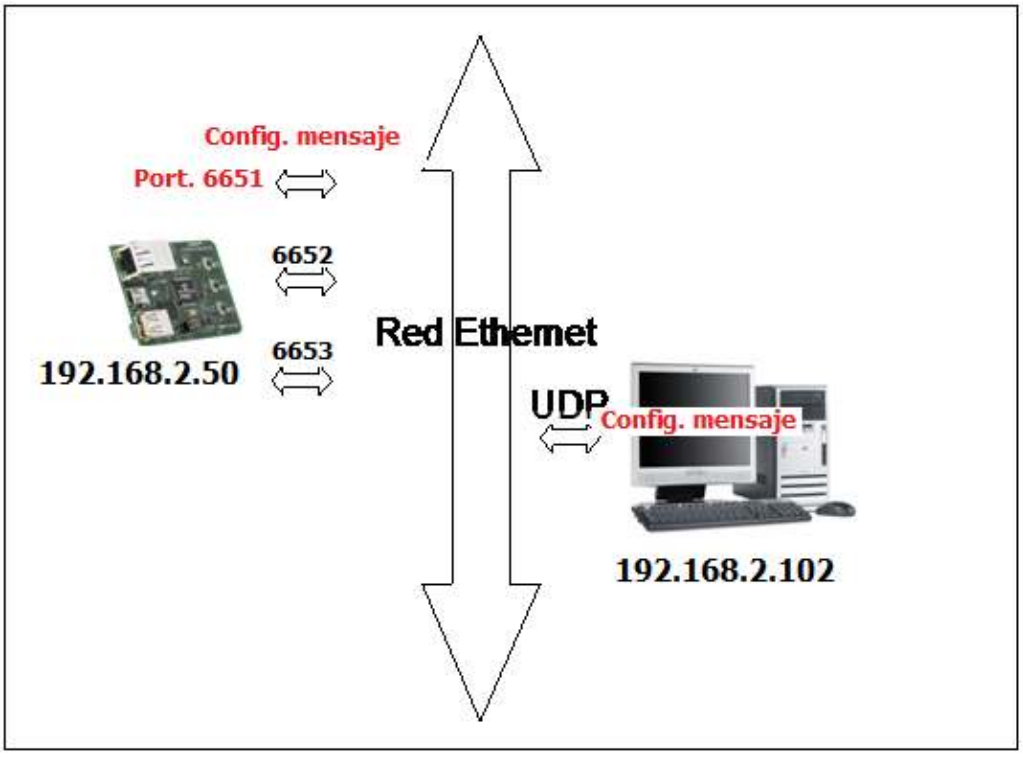


Figura 5.5 Propuesta de conectividad UDP con servicio por puerto.

Como puede verse en la figura 5.5, la aplicación que reside en el nodo Ethmesh y en la aplicación que reside en la PC deben usar UDP. El nodo Ethmesh que presta el servicio podrá hacer un registro del número de puerto y de paso de la dirección IP que realiza la solicitud. De este modo el Ethmesh puede ejecutar una tarea acertada recibida por un puerto en especial, y también realizar envíos y transmisiones de paquetes utilizando UDP, lo que alivianaría el *overload* sobre la red.

5.1.7 Posibles formas de realizar la configuración de la etapa de RF

Como se mencionará más adelante, el Ethmesh emplea un dispositivo transductor para la comunicación por radiofrecuencia de altas prestaciones, el Ethmesh es capaz de aprovechar varias de las flexibilidades que el transductor ofrece realizando configuraciones remotas a través de la interfaz de red. Sin embargo, para realizar dichas configuraciones es necesario configurar una cantidad considerable de registros. En la figura 5.6 se ilustra el tipo de menú que tiene el software de prueba para el Ethmesh, una consola de Windows no es el lugar más cómodo para desplegar largas líneas de configuración, sería bastante molesto configurar cada registro en un menú tan rígido.

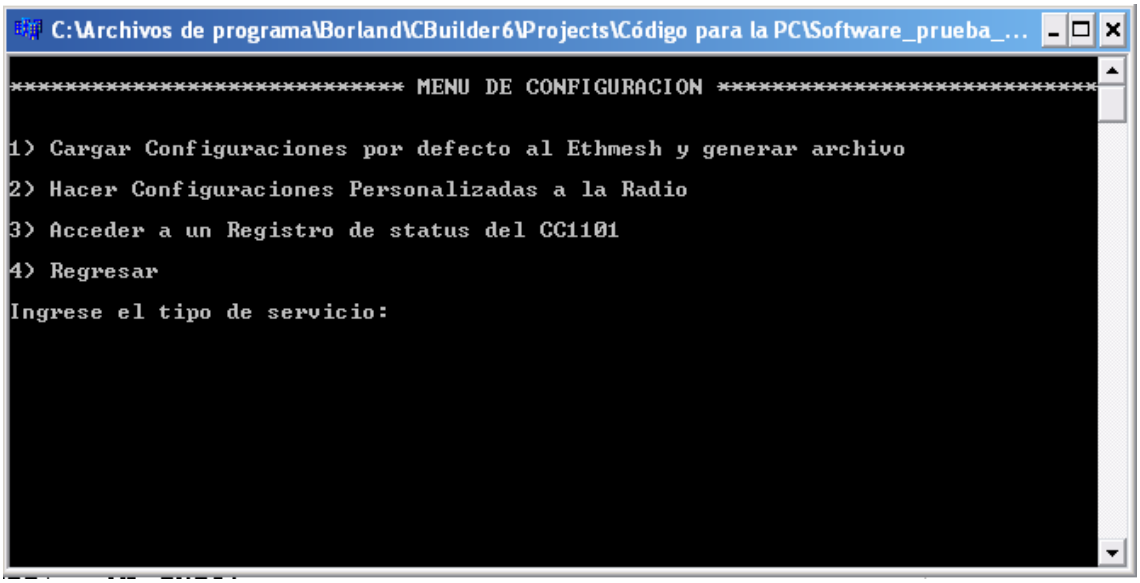


Figura 5.6 Menú de Configuración del Ethmesh.

Por esa razón se consideró utilizar una ayuda visual más cómoda para el usuario, en la figura 5.7 se muestra un fragmento del archivo de texto que se emplea para configurar de manera más eficiente los parámetros del transductor de radio. Como puede verse en la figura 5.7, un encabezado especifica algunas indicaciones para utilizar la plantilla. Lo único que el usuario debe de hacer es referirse a la hoja de datos y reescribir los valores de los registros que desea modificar, como se detalla más adelante estos registros se pueden agrupar según parámetros de la comunicación RF que éstos configuran.

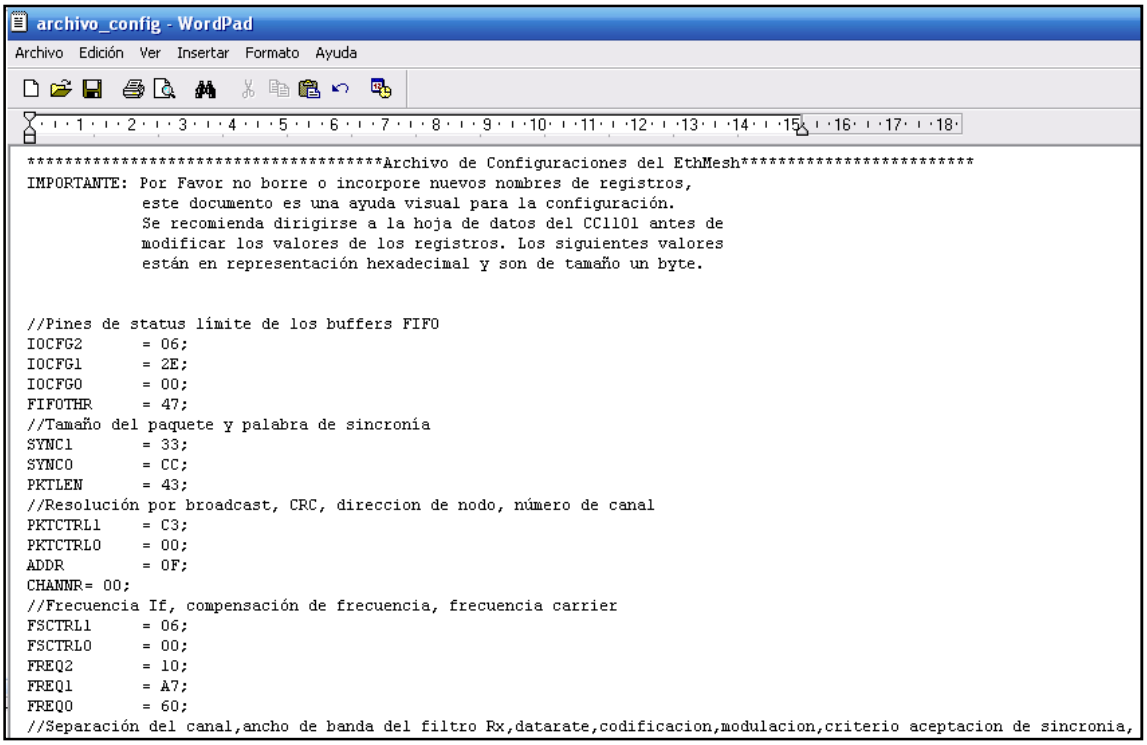


Figura 5.7 Archivo de texto para la configuración remota del CC1101.

El Ethmesh tomará los nuevos valores de este archivo de texto que reside en la ruta C:\archivo_confi, el software en la PC del Ethmesh procesa la información en este documento y envía los valores a ser cargados en los registros del transductor RF a través del Ethmesh.

5.1.8 Alternativas de hardware para la etapa de radio

El hardware utilizado para la etapa de control de flujo y configuración es un PICkit para microcontrolador de la familia de 32 bits, este hardware incluye todo lo necesario para la conexión con la interfaz de la etapa 3. Sin embargo no tiene un módulo de radiofrecuencia, por ello es necesario contemplar dos opciones. La primera opción, y la más sencilla de implementar, consiste en acoplar una placa con la etapa de radiofrecuencia al PICkit, el procedimiento que esto conlleva se explicará detalladamente en la sección de descripción de hardware de esta capitulo.

Figura 5.8 Placa de RF para acoplar el CC1101 al PICkit.

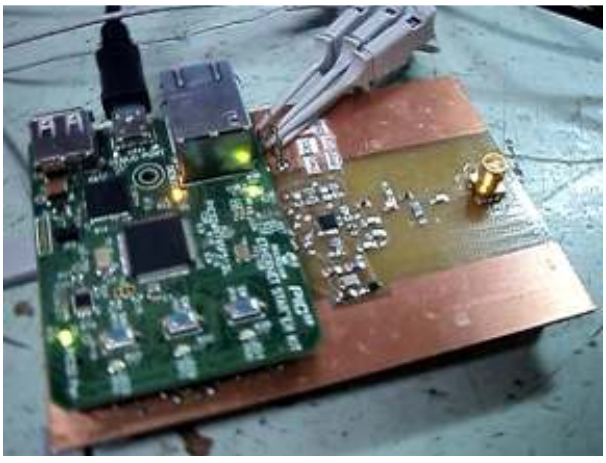


Figura 5.9 PICkit acoplado a la placa de RF.

En las figuras 5.8 se muestra la placa para la etapa de radio antes de que se le añadieran los componentes, y en la figura 5.9 se puede ver el montaje de ambas placas. La otra forma de incorporar el hardware necesario para la comunicación RF es integrando en una sola placa todos los componentes necesarios para las tres etapas del proyecto, el microcontrolador, sus componentes pasivos externos, el transductor LAN, el conector RJ45, la antena monopolo, el transductor RF y los componentes pasivos que conforman el balun para la conexión de la antena. Tal y como se muestra en la figura 5.10.

Figura 5.10 Placa completa con las tres etapas del Ethmesh.

Para evitar contratiempos se decide comenzar a trabajar con la placa de RF y el PICkit, con la idea de poder comenzar con las pruebas sin depender de la construcción de un hardware mucho más complicado. Al final se probó todo el funcionamiento con este dispositivo, dejando a un lado la propuesta que integra las tres etapas. Este hardware no fue posible finalizar por la falta de un componente que no se conseguía durante los primeros meses del proyecto. Para cuando nuevamente aparece en venta, por parte de los distribuidores a los que el DIEC compra componentes, el distribuidor del transductor LAN (DP8384C de National), ofrece un tiempo de entrega mayor al disponible para completar el proyecto. Sin embargo, el PCB fue diseñado (ver apéndice A.4) y la placa se construyó casi en su totalidad, abriendo la posibilidad a que otras personas retomen el proyecto para implementarle más funcionalidades y completar el hardware.

5.2 Descripción del hardware

5.2.1 Descripción general del hardware empleado

En esta primera sección se presenta un diagrama de bloques que incluye todos los elementos fundamentales que conforman el dispositivo, en secciones posteriores se destacarán los componentes medulares y se plantean todos los detalles pertinentes para cada uno: criterio para la selección, funciones, operación y conexión entre otros que se consideren pertinentes mencionar.

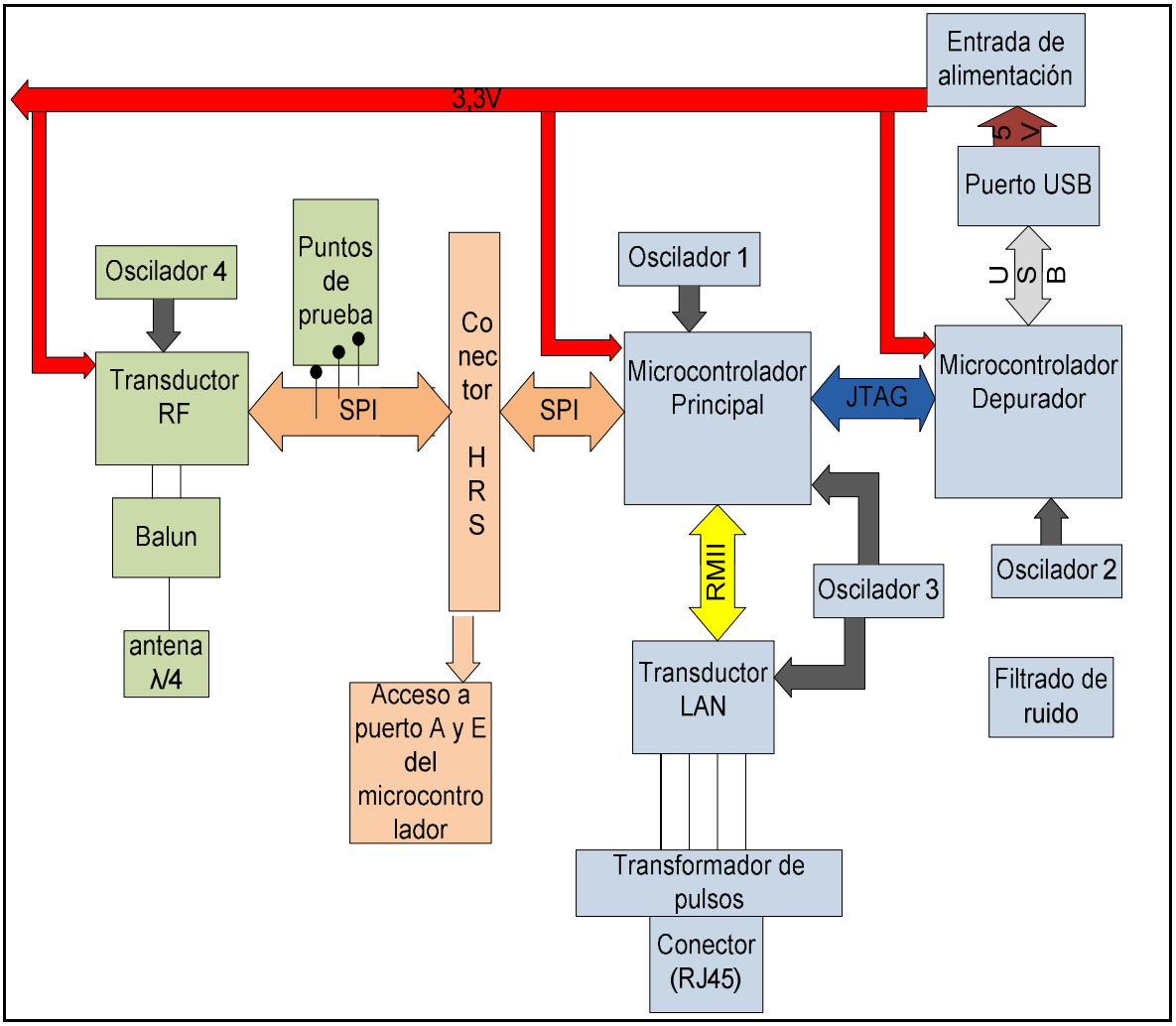


Figura 5.11 Diagrama de bloques completo del Ethmesh.

En el diagrama de la figura 5.11 se muestra todo el hardware involucrado en el desarrollo de este proyecto. Por un lado (en color celeste), se tienen los componentes incorporados en el kit de desarrollo adquirido. Como se mencionó en secciones anteriores en este capítulo, sobre este hardware se desarrollan las etapas dos y tres, es decir lo referente a las comunicaciones a través de la interfaz de Ethernet, el control principal para el flujo de los datos y la configuración de la etapa uno.

Por otro lado (en color verde) se presentan los componentes para la comunicación por radio, esta etapa de hardware requirió de su construcción en un PCB. Nótese que se especifican todas las interfaces entre cada componente, así como un bloque de filtrado de ruido que está relacionado a toda la circuitería en general.

El segundo microcontrolador de la figura 5.11 controla las comunicaciones por el puerto USB entre el dispositivo y la computadora, además controla la programación del microcontrolador principal, así como la ejecución de todas las instrucciones programadas. Para realizar dicha tarea emplea una interfaz JTAG, esta interfaz de programación y depuración permite realizar pruebas para el software programado, dichas pruebas pueden ser corridas instrucción por instrucción con el uso de puntos de parada en el código, este conjunto de herramientas de depuración resultó muy útil para descartar errores en la programación del software del Ethmesh.

5.2.2 Transductor CC1101 y diseño de la etapa de Radio Frecuencia

Actualmente se utilizan dos tipos de series de transductores para el manejo de la etapa de radio, esto se debe a que los desarrolladores de Acumine migraron hacia el CC1101 de Texas Instrument. Para la implementación de la etapa de radio se consideraron dos modelos de transductor, ambos del fabricante Texas Instrument. El uso de modelos de otros fabricantes habría sido un desperdicio de experiencia y documentación recopilada por los otros miembros del equipo.

En los anexos B2 y B3 se presentan las primeras páginas de la hoja de datos de cada transductor, como puede verse en dichos anexos, el CC1101 tiene algunas prestaciones adicionales que en el CC1000 están ausentes. El CC1101 se maneja por medio de una interfaz SPI mientras que su antecesor, utiliza una interfaz. Tal y como especifican las hojas de datos, estos transductores están diseñados para ser manejados por un microcontrolador, el manejo a través de un micro será más sencillo si se utiliza una interfaz de comunicación SPI, debido a que es un estándar de comunicación se cuenta con mayor cantidad de referencias. Por otro lado, la escritura y lectura de datos desde el transductor se lleva a cabo por medio de dicha interfaz. Entones el flujo de los datos microcontrolador-transductor será más rápido utilizando la interfaz del CC1101, esto representa una buena razón para optar por el modelo más moderno.

Otro aspecto muy importante a considerar en el diseño de la etapa de radio está relacionado al uso del espectro electromagnético. Considerando que en la actualidad muchos países cuentan con regulaciones y leyes para controlar el uso de las distintas bandas de radiofrecuencia, se ha seleccionado el transductor CC1101, ya que está diseñado para operar en un rango ubicado entre 300-348MHz, 387-464MHz y 779-928MHz. O sea que opera en bandas designadas para dispositivos de corto alcance (SRDs), las cuales gozan de uso absuelto de licencia para algunas frecuencias ubicadas por debajo de 1GHz. Este rango incluye al ISM (Industrial, Scientific, Medical), estándar de frecuencias para uso médico, industrial y de propósitos científicos, cuyas bandas de uso libre por lo general son los 315MHz, 433MHz, 868MHz y 915MHz [4]. Las regulaciones más importantes para dichas bandas son EN 300 220 en Europa y la FCC CFR47 en Estados Unidos [5].

En la figura 5.12 se muestran los 20 pines del integrado.

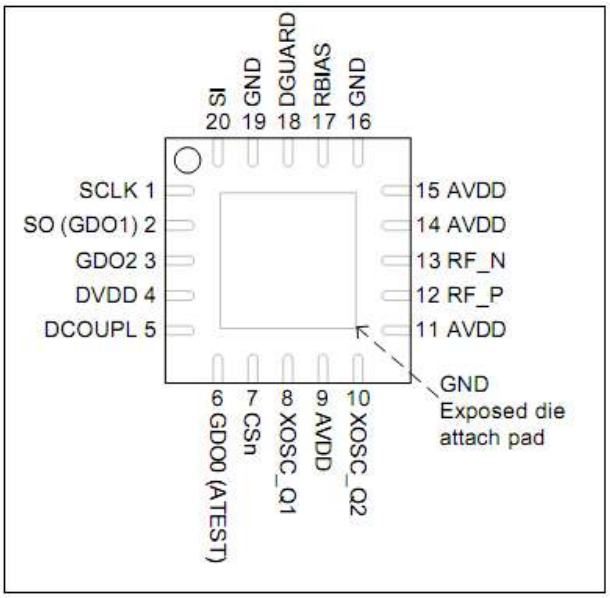


Figura 5.12 Diagrama de pines del CC1101, fabricante Texas Instruments (Tomado de [4])

Aprovechando el diagrama de la figura anterior se detallan las funciones de las principales patitas del CC1101 en la tabla 5.1.

Tabla 5.1 Descripción de los pines del CC1101.

Número	Nombre del Pin	Tipo de Pin	Descripción
1,2,7,20	SCLK, SO(GDO1), CSn, SI.	Salidas y entradas digitales	Comunicación SPI
3 y 6	GDO0, GDO2	Salidas digitales	Salida para pruebas y control de flujo.
12 y 13	RF_P, RF_N	Salida/entrada de RF	Salidas o entradas positivas y negativas en los modos transmisión y recepción.

El integrado cuenta con pines para la comunicación con el microcontrolador, señales de estatus y prueba, entrada/salida para la antena, entrada del cristal, alimentación analógica-digital y tierras.

En a) de la figura 5.13 se presenta un diagrama de bloques del integrado, las señales de RF recibidas son amplificadas por amplificador de bajo ruido (LNA), y llevadas en cuadratura (I y Q) a una frecuencia intermedia (IF). En esa frecuencia las señales I/Q son digitalizadas. El control de la ganancia el filtrado fino de canal y la demodulación son realizados digitalmente. La parte de transmisión del CC1101 se basa en la síntesis directa de la frecuencia RF. El sintetizador de frecuencia incluye un oscilador controlado por voltaje y un desfasador de 90 grados para generar las señales I/Q hacia los mezcladores de heterodinaje en el modo de recepción. Un cristal oscilador genera la frecuencia para el sintetizador y para los pulsos para los ADC´s y la parte digital. La banda base digital permite la configuración del canal, el manejo de los paquetes y el acceso a los buffers. Para la aplicación a desarrollar a 433MHz se toma la configuración sugerida en la hoja de datos que se presenta en b) de la figura 5.13.

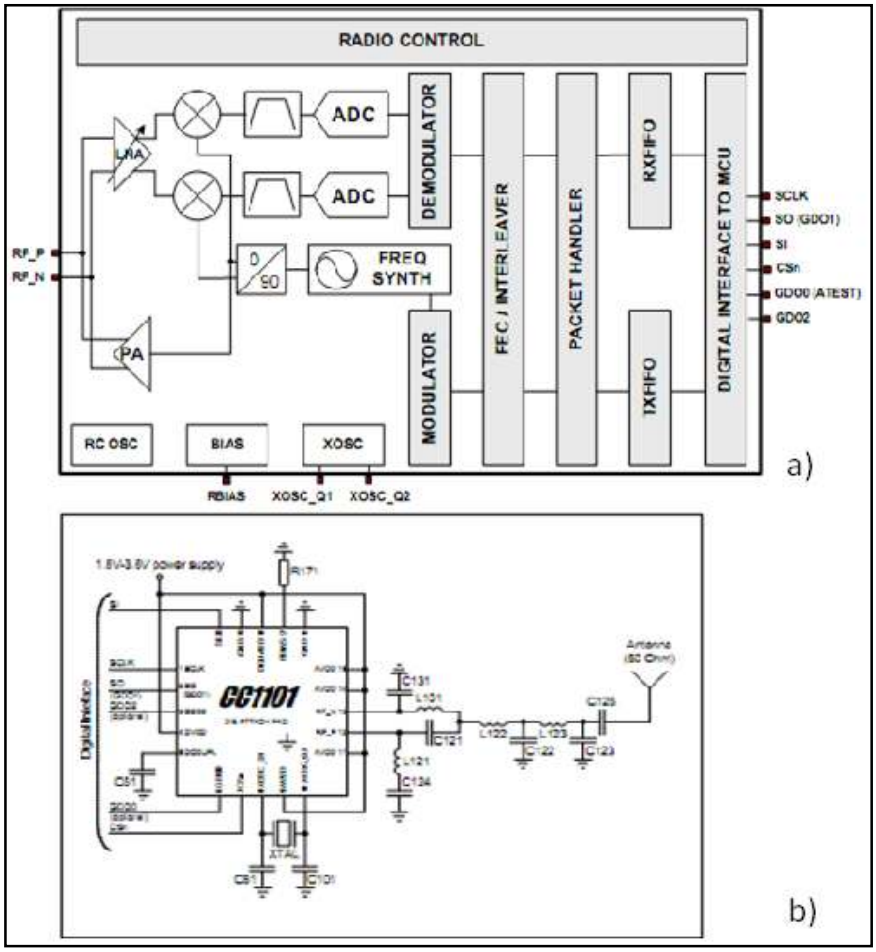


Figura 5.13 Diagrama de bloques y circuito típico de aplicación. (Tomado de [4])

Como se observa en b) de la figura 5.13, la salida y entrada RF balanceada del CC1101está diseñada para utilizar un acoplamiento sencillo y económico así como una red de desbalance implementada sobre el PCB (Onboard). Esta red convierte la señal diferencial entre los pines RF_P y RF_N, en una señal simple que se puede conectar a una antena monopolo de λ/8 Para dimensionar los componentes pasivos, se recurre a las notas de aplicación ofrecidas por el fabricante [2].

Ahora que se han revisado los aspectos más generales de este integrado, es necesario profundizar un poco más en su comportamiento y funciones ya que gran parte del tiempo dispensado fue para realizar el manejo de este transductor. Además se consideró que si se dejan claros algunos conceptos en la descripción del funcionamiento de este hardware, entonces se hará más sencilla la comprensión del software.

Las configuraciones más esenciales para el CC1101 incluyen el modo de transmisión y recepción, la selección de un canal de RF, el establecimiento de un data rate, un formato de modulación, un ancho de banda para el filtrado de la entrada, una potencia de transmisión y el almacenamiento de los datos en FIFOs separadas para transmisión y recepción de tamaño 64 bytes. Todos esos parámetros son definidos por medio de la interfaz SPI, por ello es importante comenzar por entender cómo funciona esta interfaz para el CC1101.

5.2.2.1 Interfaz SPI para el acceso a registros y datos

Como ya se ha mencionado antes la interfaz entre el microcontrolador se lleva a cabo mediante una conexión SPI con transmisiones de un byte. Toda transacción comienza con un byte de encabezado que se compone de 6 bits menos significativos para la dirección de registro, un séptimo bit que designa el tipo de acceso y el octavo que identifica si se trata de una lectura o una escritura. La selección de chip (CS) debe estar en cero para cada transmisión. Toda transmisión SPI es bidireccional, esto quiere decir cuando el controlador

envía el encabezado al CC1101 recibe al mismo tiempo un byte por la entrada SI, dicho byte es conocido como byte de status. En este byte de status el transductor reporta información sobre el status de la máquina de estados del CC1101, la cantidad de bytes disponibles en el FIFO de recepción o transmisión y la disponibilidad del chip.

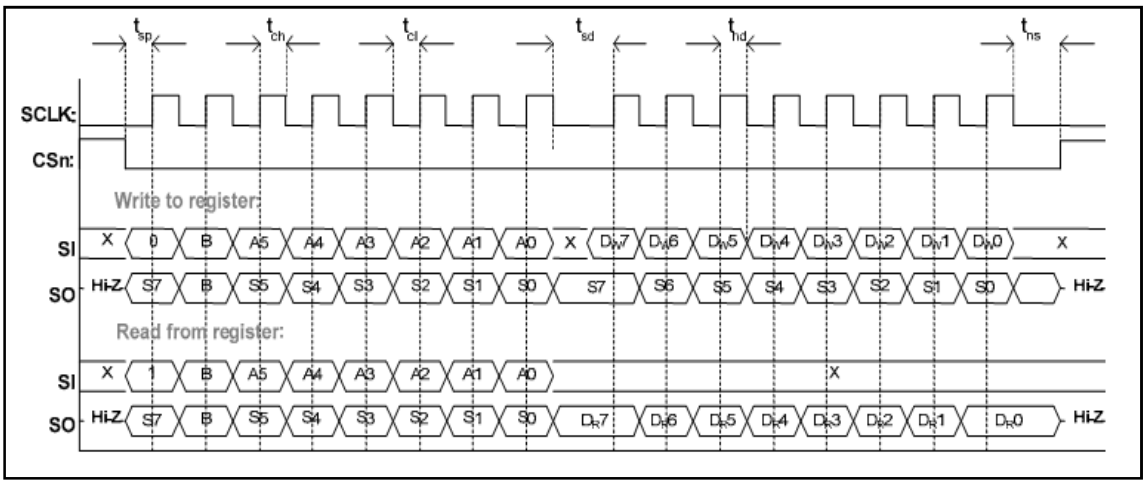


Figura 5.14 Diagrama de tiempos para operaciones de lectura y escritura. (Tomado de [4])

En la figura anterior se muestra primero una operación de escritura al CC1101, cuando el control le envía el primer byte (que contiene la indicación de escritura, el identificador de tipo de acceso y la dirección del registro a configurar), el CC1101 responde con un byte de status que contiene la información resumida en la tabla 5.2; durante las siguientes escrituras el transductor continúa enviando el status byte por cada escritura que se haga al o los registros. En otro caso, cuando el control realiza una lectura de algún registro del transductor también envía un primer byte de encabezado con el bit 7 en uno para indicar lectura, entonces el CC1101 responde de nuevo con un byte de status y posteriormente con los datos solicitados por el control.

Tabla 5.2 Descripción de los pines del CC1101.

<table>
<tr><th>Bits</th><th>Nombre</th><th>Descripción</th></tr>
<tr><td>7</td><td>CHIP_RDYn</td><td>Se mantiene en uno mientras el cristal y la alimentación no se hayan estabilizado</td></tr>
<tr><td>6:4</td><td>STATE [2:0]</td><td><table><tr><td>Valor</td><td>Estado</td></tr><tr><td>000</td><td>IDLE</td></tr><tr><td>001</td><td>RX</td></tr><tr><td>010</td><td>TX</td></tr><tr><td>011</td><td>FSTXON</td></tr><tr><td>100</td><td>CALIBRATE</td></tr><tr><td>101</td><td>SETTLING</td></tr><tr><td>110</td><td>RXFIFO_OVERFLOW</td></tr><tr><td>111</td><td>TXFIFO_UNDERFLOW</td></tr></table></td></tr>
<tr><td>3:0</td><td colspan="2">Número de bytes disponibles en la FIFO de Rx o Tx.</td></tr>
</table>

Los estados indicados en la tabla anterior están relacionados a la máquina de estados que controla el funcionamiento del CC1101 y se especifican más adelante. Cuando se realizan funciones de lectura los bits 3:0 indican la cantidad de bytes restantes para la lectura en el buffer de recepción y en caso de realizar una escritura informan sobre la cantidad de bytes disponibles para escribir en el buffer de transmisión, este valor sólo cambiará cuando el número de bytes disponibles para leer o para escribir sea menor a 16.

Existen dos tipos de acceso a los registros internos del transductor, el acceso simple y el acceso "burst". En el acceso simple cada modificación o lectura de un registro implica el envío de un primer byte de encabezado para direccionar el registro a acceder, una vez direccionado el registro se escribe un valor o se lee, en el caso de lectura es necesario hacer el envío de un byte "dummy" que en realidad no es importante ni afectará la operación del CC1101; su único fin es el de enviar algo por la patita de SO mientras se realiza la lectura del registro en interés.

Por otro lado el acceso "burst" requiere únicamente de un direccionamiento inicial, ya que luego hace lecturas o escrituras continuas de los registros consecutivos. Quiere

decir que los registros que se quieren acceder deben ser consecutivos, la operación de lectura o escritura se detendrá haciendo que el "CS" sea uno de nuevo. La figura 5.15 ayuda a entender la operación. Las operaciones de burst o simples se distinguen mediante el séptimo bit, cuando éste es cero la operación es sencilla, si vale uno la operación es con burst.

Existen tres tipos de registros que incluyen configuración, estatus y "storbes", los strobes son instrucciones de tamaño un byte que el controlador puede enviar al CC1101 para moverlo por sus distintos estados. Los strobes y las configuraciones se direccionan desde 0x30 hasta 0x3D, éstos se distinguen entre sí con el burst bit. En el caso de un strobe el burst siempre es uno.

El tamaño de los buffers es 64 bytes, hay una FIFO para lectura y otra para escritura, sus direcciones comienzan en 0x3F, dependiendo del byte de lectura/escritura se identifica cuál buffer es manipulado. La figura 5.15 ejemplifica cada caso mencionado anteriormente.

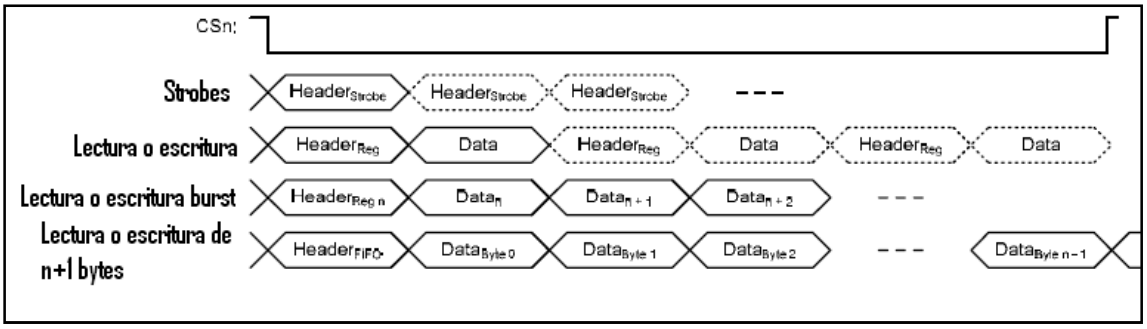


Figura 5.15 Diagrama de tiempos para varias operaciones. (Tomado de [4])

Adicional a los 4 pines del SPI se tienen dos conexiones más que serán utilizadas para generar las notificaciones de recepción y fin de transmisión de paquetes, estos pines son programables y pueden tener múltiples funciones sin embargo, para simplificar las cosas solo serán usados para generar eventos hacia el control. La tabla 5.3 muestra algunas de las 63 funciones que se pueden asignar.

Tabla 5.3 Descripción de algunas funciones programables para los pines GDOx.

GDOx_CGF[5:0]	Descripción
0x00	Asociada al RX FIFO, asertiva cuando la FIFO se llena por arriba de un nivel umbral. Desacierta cuando RX FIFO se vacía por debajo del umbral.
0x01	Asociada al RX FIFO, asertiva cuando la FIFO se vacía por debajo de un nivel umbral. Desacierta cuando RX FIFO se vacía por completo.
0x02	Asociada al TX FIFO, asertiva cuando la TX FIFO se llena por arriba de un nivel umbral. Desacierta cuando TX FIFO se vacía por debajo del umbral.
0x03	Asociada al TX FIFO, asertiva cuando la TX FIFO se llena por completo. Desacierta cuando TX FIFO se vacía por debajo del umbral.
0x04	Asertiva cuando RX FIFO ha hecho overflow, desacertada cuando se hace un flush de la RX FIFO.
0x05	Asertiva cuando TX FIFO ha hecho underflow, desacierta cuando se hace un flush de la TX FIFO.
0x06	Asertiva cuando se ha detectado una palabra de sincronización y desacierta en el fin de paquete. En RX también desacierta cuando el paquete es descartado.
0x07	Asertiva cuando se ha recibido un paquete que aprueba el CRC.
0x08	Asertiva cuando se tiene una calidad de preámbulo aceptable, quiere decir cuando PQI está por arriba del PQT programado.
0x09	Asertiva cuando el RSSI está por debajo de un nivel umbral.

5.2.2.2 Data rate y ancho de banda del filtro de entrada

La velocidad de transmisión de los datos se puede evaluar en términos del bit rate o del symbol rate, en este caso se utilizará una codificación Manchester, como se menciona en el marco teórico, una codificación de este tipo reduce la velocidad de transmisión a la mitad, debido a que cada bit será representado por dos símbolos. Lo que se programa en los registros MDMCFG3.DRATE_M y el MDMCFG4.DRATE_E es realmente la velocidad de los símbolos, a pesar de que en las hojas de datos se refieren a este término como data rate se manejó una ambigüedad que fue resuelta tras realizar algunas transmisiones y se hacer algunas mediciones, dichos resultados se plantean en el siguiente capítulo. Para determinar el valor a cargar en los registros el fabricante proporciona las siguientes ecuaciones:

$$R_{DATA} = \frac{(256+DRATE\_M)2^{DRATE\_E}}{2^{28}} f_{XOSC} \tag{5.1}$$

De donde pueden realizarse las siguientes dos aproximaciones,

$$DRATE\_E = \left\lfloor \log_2 \left(\frac{R_{DATA}\, 2^{20}}{f_{XOSC}} \right) \right\rfloor \tag{5.2}$$

$$DRATE_M = \frac{R_{DATA}\, 2^{28}}{f_{XOSC}\, 2^{DRATE\_E}} - 256 \tag{5.3}$$

Entonces se pueden despejar los valores para ajustar velocidades desde 0.6KBaud hasta 500kBaud con un distanciado entre ellas mínimo según la tabla 5.4.

Tabla 5.4 Posibles velocidades de símbolo para el CC1101.

Min. Data Rate [kBaud]	Data Rate Típico [kBaud]	Máx. Data Rate [kBaud]	Mínimo tamaño de paso [kBaud]
0.6	1.0	0.79	0.0015
0.79	1.2	1.58	0.0031
1.59	2.4	3.17	0.0062
3.17	4.8	6.33	0.0124
6.35	9.6	12.7	0.0248
12.7	19.6	25.3	0.0496
25.4	38.4	50.7	0.0992
50.8	76.8	101.4	0.1984
101.6	153.6	202.8	0.3967
203.1	250	405.5	0.7935
406.3	500	500	1.5869

De las velocidades en la tabla 5.4 se utilizó la de 38.4 kBaud, para ello se programaron los valores indicados en el apéndice A.4, que muestra una tabla con todos los valores programados para cada registro del CC1101. Entonces se puede deducir que la velocidad de transmisión en bits será de 19.2 Kbps, en capítulos posteriores se retoman estos resultados y se relacionan con el ancho de banda de la señal FSK obtenida.

Conocer el ancho de banda de la señal es importante para evaluar el uso del espectro, prevenir interferencias con otros equipos y seleccionar los filtros adecuados. El CC1101 dispone de un filtro de entrada con un ancho de banda programable, los valores deben cargarse en dos registros (ver A.4), MDMCFG4.CHANBW_E y MDMCFG4.CHANBW_M, con éstos valores se hace un escalado de la frecuencia del oscilador. La tabla 5.5 muestra una lista de los posibles valores para el filtro de entrada para este caso en que se usó un cristal de 26MHz. Para obtener el mejor rendimiento del filtro se debe ajustar de forma que el ancho de banda de la señal a trabajar sea el 80% del ancho de banda del filtro de entrada, esto asegura que el filtro de entrada no es muy ajustado y puede lidiar con desviaciones en la frecuencia central.

Tabla 5.5 Anchos de banda para el filtro de entrada, en kHz.

MDMCFG4. CHANBW_M	**MDMCFG4.CHANBW_E**			
	00	**01**	**10**	**11**
00	812	406	203	102
01	650	325	162	81
10	541	270	135	68
11	464	232	116	58

El valor que mejor se ajusta según el ancho de banda de la señal esperada es el de 162 kHz, por ende el registro MDMCFG4 se programa con el valor hexadecimal 0x9A, este valor fue estimado con base en la ecuación 3.7 y considerando que la señal a recibir debía ser del 80% del BW del filtro (ver anexo A.4 y en resultados y análisis).

5.2.2.3 Sincronización y manejo de los paquetes.

La sincronización de los bits se lleva a cabo tomando el reloj desde las señales incidentes, para ello es necesario que se haya predeterminado una velocidad para los símbolos, este dato se toma de la programación de un symbol rate en los registros mencionados en la sección pasada. Cada cierto tiempo se realiza una re sincronización para corregir errores de sincronía con las señales entrantes.

Para sincronizar los bytes se ejecuta una búsqueda continua de una palabra de sincronización. La palabra de sincronización es un valor de 16 ó 32 bits insertado automáticamente por el modulador al inicio de cada paquete durante una transmisión. Además el demodulador utiliza este valor para delimitar cada byte en el stream de bits.

Además la palabra de sincronización funciona como identificador, ya que solo los paquetes con la correcta palabra de sincronización serán aceptados por el dispositivo, el grado de seguridad puede aumentar o disminuir eligiendo si se desea la coincidencia completa o si se tolera un error en un bit, también es posible elegir entre palabras de 16 ó 32 bits de largo. En este caso se utilizó una palabra de 16 bits con un valor hexadecimal 0xCC33 y con una exigencia de 16/16, o sea que no se tolera errores en ningún bit, de esta forma se asegura que la sincronización ha sido exitosa y hay una baja posibilidad de errores en el paquete.

En general el CC1101 contiene muchas opciones para el manejo de los paquetes, esta facilidad permite al transductor tomar decisiones por hardware automáticamente liberando de muchos procedimientos de control al microcontrolador lo cual pareciera una clara ventaja sin embargo, darle tanto control al CC1101 para manejar todo lo correspondiente a la recepción de paquetes no es tan conveniente en este caso, por ello se utilizaron solo algunas de todas las condiciones que puede manejar por sí mismo el transductor. Para una transmisión el CC1101 puede agregar los siguientes bytes de control:

- Una cantidad programable de bytes para el preámbulo.
- Una palabra de sincronización de uno a dos words de tamaño.
- Un word del cálculo del CRC computado sobre la carga efectiva.

Se decide utilizar una palabra de sincronización de 2 bytes y un preámbulo de 24 bytes. El preámbulo es una señal alternada de unos y ceros lógicos cuyo fin facilitar la sincronización inicial de bits para el demodulador, de manera que cuando el nodo es llevado al modo de recepción por el control, éste comienza a buscar un preámbulo válido.

El método para dar validez al preámbulo consiste en contar la cantidad de unos y ceros detectados de manera alterna, conforme acumule una cierta cantidad de conteos entonces se considera como preámbulo válido; pero si se repite un cero o un uno entonces se rebajan ocho conteos al acumulado. Este acumulado es conocido como indicador de calidad de preámbulo (PQI), cuando esta cantidad alcanza o sobrepasa a un umbral 4·PQT, entonces se procede a la detección de palabra de sincronía. El PQT es un valor programable mediante uno de los registros de configuración de paquete.

Es posible configurar una de las salidas GDOx para generar un evento en el momento que se reciba un preámbulo aceptable. Para este diseño se utilizó un preámbulo de 26 bytes, el valor para PQT se programó inicialmente en 0x07, lo que quiere decir que es necesario acumular 28 bits alternos sin repetirse para aceptar el preámbulo. Este criterio fue programado nuevamente tras tomar algunas mediciones y considerarlo muy rígido, el PQT se establece al final en 0x03, de manera que son necesarias solo 12 cuentas para aceptar el preámbulo. La decisión se toma debido a que la palabra de sincronización ofrece un segundo filtro para aceptar paquetes durante las recepciones, además para reafirmar la veracidad del dato también se tiene un CRC, entonces se consideró innecesario un criterio tan estricto. A pesar de haber bajado la exigencia para el control de calidad del preámbulo, no se disminuye la cantidad del mismo por una sencilla razón: se necesita que entre cada transmisión haya un retraso considerable ya que las cajas se ponen en modo de recepción por periodos pequeños de tiempo cada segundo. Esto quiere decir que si un determinado nodo cerca de otro comienza a transmitir pero su preámbulo no es detectado porque el destino no se encuentra en escucha, entonces aunque el nodo se pongan en recepción en el momento que el emisor está enviando la palabra de sincronía, ésta no será tomada ya que nunca se hizo la sincronización de bit. Las configuraciones para el tamaño y evaluación del preámbulo pueden verse en la tabla del anexo A.4.

En la figura 5.16 se observa la estructura que debe tener el paquete para ser manejado de manera automática por hardware en el CC1101.

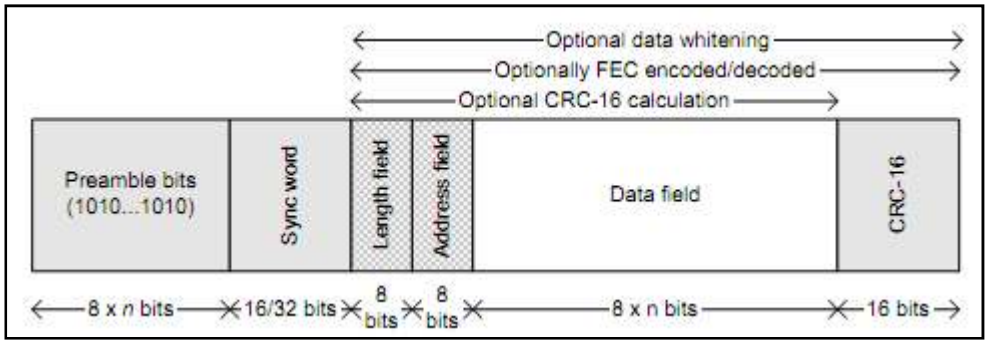


Figura 5.16 Estructura de un paquete según el CC1101.

Se muestran además en la parte superior de la imagen todas las opciones que se pueden agregar para añadir seguridad a los paquetes así como para darles mantenimiento. Pero durante los inicios del proyecto Acumine se trabajó con otros dispositivos para la parte de la red inalámbrica, de allí que inevitablemente se heredaran algunas características, entre ellas la estructura de los paquetes, quiere decir que los encabezados de los paquetes difieren en algunos detalles de la forma presentada en la figura 5.16. Los campos de dirección y tamaño se encuentran, por ende no se puede emplear el descarte de paquetes automático ofrecido por el transductor, la solución para esta situación es colocar un valor de 0xFF en ambos campos y configurar el CC1101 para trabajar con direcciones de broadcast, entonces todos los paquetes llevan como dirección destino 0xFF de manera que el CC1101 recibe todos los paquetes y los descarta posteriormente por software. Para el diseño del Ethmesh este método es más conveniente ya que se debe tener en cuenta que el Ethmesh se involucra en la red de 433MHz utilizando varios números de nodo (llamados también máscaras).

Con respecto al tamaño de los paquetes, no se realiza verificación del tamaño del paquete por hardware, este parámetro se manejó por software utilizando espacios configurables dentro de la carga efectiva. La función de CRC tampoco se deja en manos del CC1101, sino que se realiza por parte del control luego de procesar el paquete. En este caso el micro verifica que el CRC contenido en el paquete y el calculado con base en la carga efectiva coincidan para cada paquete. Como ya se ha mencionado la detección de preámbulo y de palabra de sincronización si se han dejado en manos del transductor ya que

estas funciones son insertadas y retiradas automáticamente por el CC1101, o sea que su funcionamiento se mantiene en la capa de hardware.

El manejo de los paquetes se encuentra muy relacionado a la estructura en sí del paquete, por esta sección se discute sobre la forma que finalmente se decidió para el paquete, en la figura 5.17 se realiza una comparación entre la estructura esperada por el CC1101 para el paquete y la forma que finalmente se le dio.

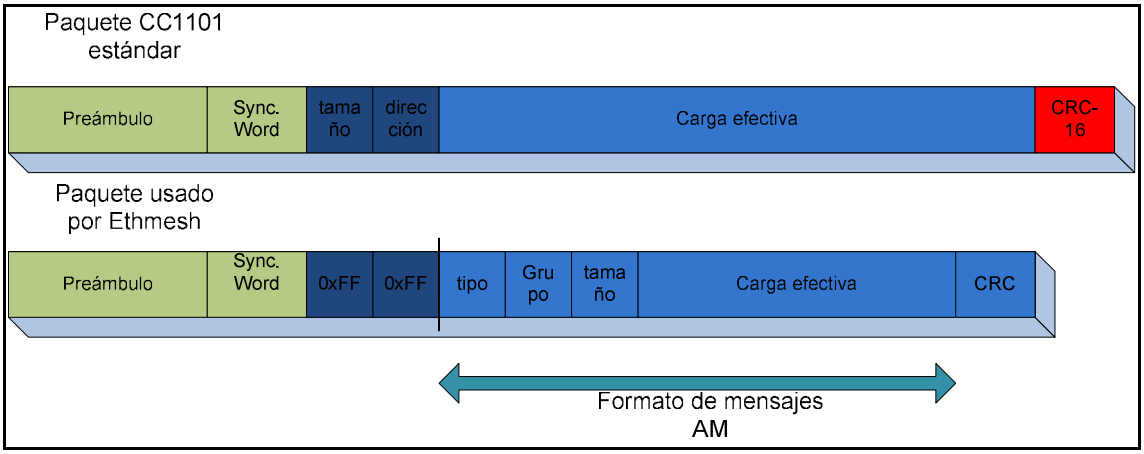


Figura 5.17 Estructura de un paquete según el CC1101.

El preámbulo, la palabra de sincronización y el chequeo por redundancia cíclica son agregados para una transmisión y retirados en una recepción por el hardware del C1101, pero debido a que no se utilizó el CRC por hardware, el paquete usado por el Ethmesh es dos bytes más pequeño. En azul se muestran los datos que son definidos por el software, en azul oscuro se ilustran los campos en los cuáles fue necesario colocar 0xFF para obtener un broadcast y descarte por software, para poder realizar dicha acción es necesario tomar alguno de los campos de la carga efectiva para colocar una dirección destino. El formato Am se refiere a la estructura del mensaje utilizado por las cajas Acumine para la comunicación en la banda de baja frecuencia, el campo de tipo distingue entre mensajes de alerta, advertencia o ping. El grupo es un identificador extra para saber si se trata de un camión, vehículo liviano, entre otros. Finalmente el tamaño brinda información acerca de

la cantidad de bytes contenidos únicamente en la carga efectiva, por ende el tamaño total del paquete contenido en el buffer de recepción del CC1101 es igual a la carga efectiva más siete bytes de control.

El transductor utilizado puede manejar protocolos con paquetes de tamaño variable y fijo. Para ambos casos puede trabajar con paquetes de hasta 256bytes, de lo contrario es necesario utilizar una aplicación de transmisión de paquetes infinitos. Sin embargo la cantidad de datos a manejar entre la red cableada y la inalámbrica no es masiva por o cual se ha decidido establecer 256 bytes como tamaño máximo.

Se utilizó un tamaño de los paquetes fijo, pero que es configurable por medio del puerto de servicios del Ethmesh. Este tamaño de paquete no será utilizado como criterio de descarte pero, se utilizará para definir la cantidad de bytes en las lecturas y escrituras hacia las memorias FIFO de transmisión y recepción. El algoritmo empleado para recibir y enviar los datos, que se explicará en la descripción del software, toma un valor de referencia pre configurado con máximo de 256 y recibe o envía paquetes de cualquier tamaño preestablecido. Dicho tamaño debe ser establecido previamente en uno de los registros de configuración del transductor ya que es utilizado para generar las interrupciones de fin de paquete que permiten llevar a cabo las recepciones y transmisiones.

El CC1101 cuenta con dos memorias para almacenar los datos a transmitir y recibir, dichas memorias son de tipo FIFO (el primer byte en entrar es el primero en salir), de tamaño 64 bytes. El microcontrolador necesita darse cuenta cuando un paquete ha sido recibido o transmitido, adicionalmente para paquetes mayores a 64bytes es necesario realizar lecturas de la FIFO_RX mientras se encuentra recibiendo y la TX_FIFO debe ser escrita en medio de una transmisión. Esto significa que el control necesita saber la cantidad de bytes que puede leer o escribir, para ello se tienen dos posibles soluciones:

- La primera sugiere hacer un chequeo constante del byte de estatus, esto permite enterarse cuando se tiene una cantidad menor a 16 bytes en cualquiera de las dos memorias. Esto viene siendo una clase de "polling" del byte de respuesta para cada byte que se escribe o se lee.

- La otra alternativa es utilizar la generación de eventos como interrupciones al flujo constante del microcontrolador para que éste realice las lecturas o recargas necesarias a las FIFO del CC1101.

La alternativa seleccionada fue por medio de interrupciones, esta opción fue seleccionada debido a que el hacer polling a través de la interfaz SPI genera ciertos errores [38].

5.2.2.4 Memorias FIFO del CC1101

El control interno del CC1101 puede detectar fallas en el uso de las memorias, puede detectar "overflow" o sobre escrituras en la FIFO_RX y "underflow" o no escrituras en la FIFO_TX. Por otro lado el microcontrolador es responsable de no realizar más lecturas de las disponibles desde la FIFO_RX y no exceder las escrituras en el FIFO_TX. La cantidad de bytes en las memorias puede ser conocido accediendo a un registro de estatus.

Es posible programar un nivel de umbral para las escrituras en la FIFO_RX y uno para lecturas de la FIFO_TX. De manera que conforme el CC1101 recibe los bytes de un paquete y los va escribiendo en la memoria FIFO_RX se va haciendo una comparación con el nivel de umbral programado, si ese nivel se excede entonces se genera un evento. Dicho evento puede ser usado como interrupción para el microcontrolador, si en dicha interrupción se realiza la lectura de la FIFO_RX entonces se evita que el control del CC1101 sea empujado hacia el estado de overflow. En la tabla5.6 se muestran los posibles valores para el umbral (threshold), un solo umbral es programado para amas memorias.

Tabla 5.6 Posibles umbrales para las memorias y su nivel de llenado.

Umbral de la FIFO	Bytes en TX_FIFO	Bytes en RX_FIFO
0000	61	4
0001	57	8
0010	53	12
0011	49	16
0100	45	20
0101	41	24
0110	37	28
0111	33	32
1000	29	36
1001	25	40
1010	21	44
1011	17	48
1100	13	52
1101	9	56
1110	5	60
1111	1	64

Para este diseño se utilizó la configuración “0111”, la cual permite que se llene la memoria RX_FIFO hasta completar 32 bytes, y permite que se envíen los bytes en la TX_FIFO hasta igualar a 33 bytes. Para estos eventos se programó una salida de control GDOx de manera que el microcontrolador puede proceder a la lectura de la FIFO en cada recepción, o continuar con la escritura de los bytes en la FIFO para completar una transmisión.

Fue importante revisar las erratas para realizar consideraciones en esta etapa, ya que la memoria FIFO_RX no debe ser vaciada por completo mientras es escrita al mismo tiempo, esto implica que las lecturas en medio de la recepción de un paquete no pueden leer la cantidad total equivalente al umbral. Para este diseño se realizan lecturas de 31 bytes cada vez que se genera el evento de umbral en la FIFO_RX, debido a que como se mencionó el threshold fue establecido en “0111” (ver tabla 5.6).

5.2.2.5 Formato de modulación y selección de la frecuencia.

El formato de modulación utilizado fue FSK binaria, además se utiliza codificación Manchester y por ello no es posible utilizar corrección de error FER (forware error correction), el uso de estas condiciones es incondicional debido a que son los parámetros en los que opera la banda de 433MHz, dicha frecuencia se utilizó debido a que pertenece a una banda libre de licencia (sección 5.2.2). El ajuste de la desviación de frecuencia se lleva a cabo mediante la programación del registro DEVIATN. El ajuste de este registro se hizo usando el SmartRF Studio para obtener una desviación de 31.7kHz.

La frecuencia central se ajusta mediante un valor de 24bits alocado entre los registros FREQ2, FREQ1 y FREQ0, el CC1101 permite el uso de varios canales que brincan en múltiplos de la frecuencia central. El Ethmesh utiliza un único canal ubicado en la frecuencia central. La portadora resultante está dada por:

$$f_{carrier} = \frac{f_{xsoc}}{2^{16}}\left(FREQ + CHAN\left((256 + CHANSPC\_M)2^{CHANSPC\_E-2}\right)\right) \qquad (5.4)$$

La frecuencia de operación IF está dada según la frecuencia del cristal y un valor cargado en el espacio de registro FSCTRL1.FREQ_IF.

$$f_{IF} = \frac{f_{xosc}}{2^{10}} \cdot FREQ\_IF \qquad (5.5)$$

La herramienta de software SmartRF Studio fue utilizada para encontrar la óptima frecuencia IF, con base en el espaciado entre canales y el ancho de banda del filtro de entrada.

5.2.2.6 Control interno del CC1101

El CC1101 cuenta con una secuencia lógica para realizar las tareas de recepción y transmisión, esta secuencia está controlada por una máquina de estados que realiza los

saltos de estado con base en las operaciones a realizar y en el valor pre configurado en algunos registros de control.

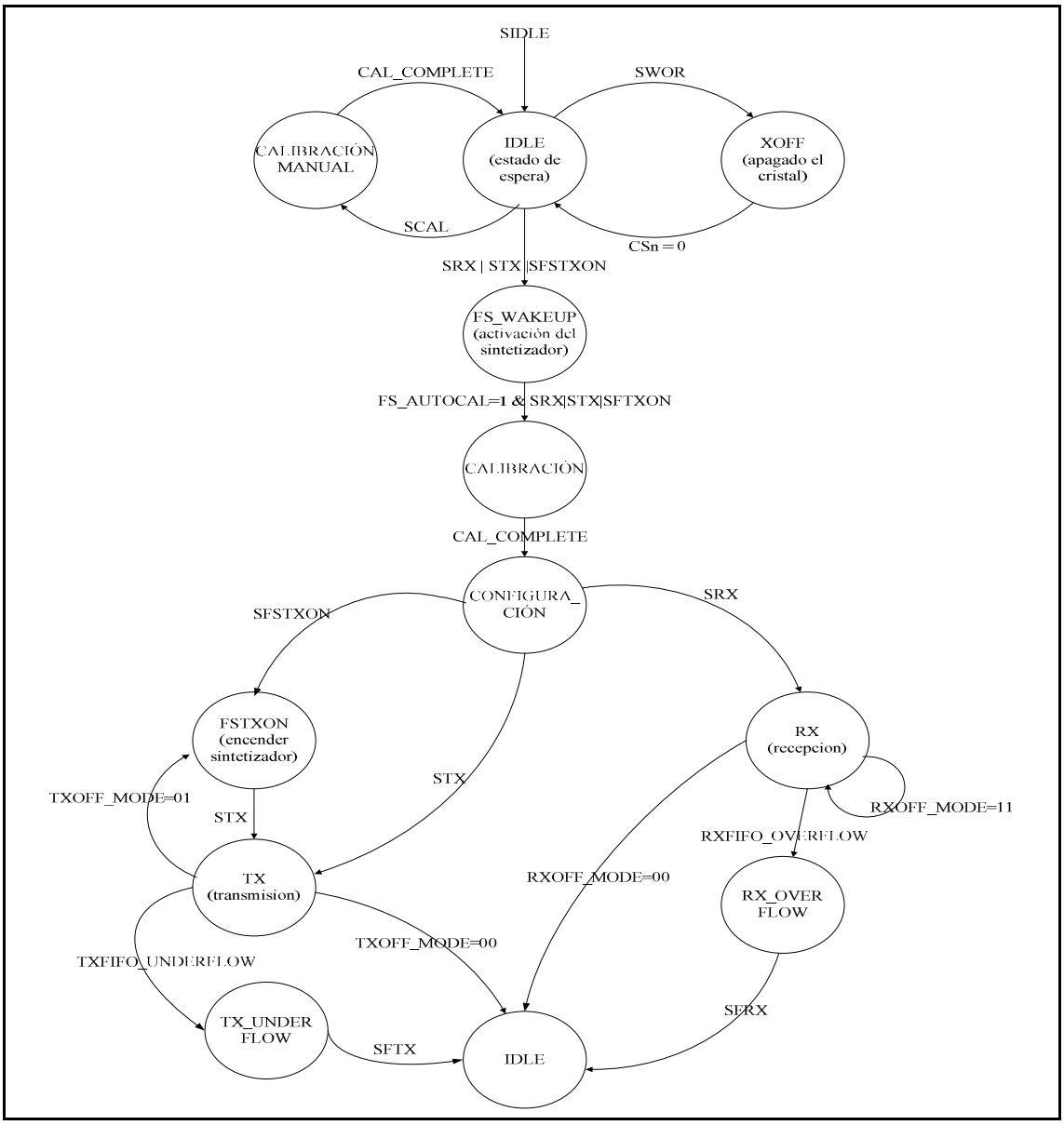


Figura 5.18 Máquina de estados simplificada del control interno del CC1101.

En la figura 5.17 se detallan las rutas entre los estados decididas para el CC1101, existen otros estados que no se presentan debido a que la figura revela una versión simplificada que considera únicamente los estados por los que se desplazará el control interno, los estados mostrados en la figura son suficientes para realizar las operaciones

requeridas para la comunicación del Ethmesh. En B.3 se presenta el diagrama de estados completo y una tabla con el detalle de los mismos.

Existen dos maneras de trazar las rutas para la máquina de estados del control del transductor, la primera es pre estableciendo valores en los registros que el CC1101 usa para determinar el estado siguiente luego de completar una tarea. La segunda forma es mediante el acceso a registros especiales denominados "strobes", los strobes son una especie de instrucciones que se envían a través de la interfaz de SPI y que el microcontrolador puede aprovechar para obligar al CC1101 a cambiar de estados según una operación requerida. Por ejemplo, para sacar al transductor de estado de espera y ponerlo a transmitir, es necesario escribir sobre la línea SDO un valor hexadecimal de 0x35, esta acción es conocida como un comando STX, la tabla 5.7 resume todos los comandos disponibles para la manipulación del control.

Tabla 5.7 Comandos para la selección de los estados del CC1101.

Dirección	Comando	Descripción
0x30	SRES	Reiniciar el chip
0x31	SFSTXON	Encender el sintetizador
0x32	SXOFF	Apagado del cristal, se enciende tras un flanco positivo en CSn
0x33	SCAL	Calibrar el sintetizador, este puede ejecutarse aun cuando se realice calibración automática.
0x34	SRX	Ir al estado de recepción, realiza una calibración previa
0x35	STX	Entrar a transmitir, pasa por la calibración y depende de si está recibiendo.

Tabla 5.7 Comandos para la selección de los estados del CC1101. (Continuación)

Dirección	Comando	Descripción
0x36	SIDLE	Sale de los estados de TX o RX, apaga el sintetizador y se va a espera.
0x38	SWOR	Envía a apagado con espera de recepción
0x39	SPWD	Entra al modo de apagado
0x3A	SFRX	Vaciar por completo la FIFO_RX.
0x3B	SFTX	Vaciar por completo la FIFO_TX
0x3D	SNOP	Ninguna operación, solo se usa para acceder al byte de estatus o recibir alguna respuesta del transceiver.

Es importante considerar que entre algunos de los estados mostrados en la figura 5.18 se encuentran muchas operaciones intermedias que si bien no se muestran ni se mencionan, sí demandan una cierta cantidad de tiempo que puede ser considerable bajo ciertas circunstancias. En la siguiente tabla se resumen algunos de los tiempos más importantes a considerar.

Tabla 5.8 Tiempos de transición entre algunos estados del CC1101.

Descripción	Tiempo de transición [µs]
IDLE a RX, con calibración	799
IDLE a TX/FSTXON, con calibración	799
TX a RX	31.1
RX a TX	30.1
TX a IDLE, sin calibración	1
RX a IDLE, sin calibración	0.1
Calibración manual	735

Un ejemplo sería: cuando el microcontrolador ejecuta un comando para enviar al dispositivo a estado de recepción y éste se encuentra en estado de espera, entonces tiene que pasar por el estado de calibración y realizar otras acciones que le demoran un tiempo total de 799µs. Si inmediatamente después de enviar el strobe por la línea SDO, el micro

realiza una lectura del registro de estado del CC1101, es muy probable que no se encuentre con el valor correspondiente a RX. Esto porque la velocidad de la interfaz de SPI es muy rápida en comparación con este tiempo de 799µs, ya que la frecuencia de la línea SCLK es 1MHz, dicha frecuencia fue seleccionada luego de revisar en la hoja de datos que el máximo valor posible que puede manejar el CC1101 es de 10MHz, pero no fue considerado pertinente usar una frecuencia tan elevada.

5.2.3 Descripción del microcontrolador y la tarjeta de desarrollo

5.2.3.1 Características generales del microcontrolador

Luego de hacer algunas consideraciones se decidió usar un microcontrolador de la marca Microchip, específicamente el modelo PIC32MXX75F256L. Este modelo presenta algunas ventajas en cuanto a disponibilidad y facilidad para conseguir documentación. Además este microcontrolador cuenta con suficiente memoria para alojar y ejecutar el código necesario, por otro lado tiene capacidad para SPI, esta interfaz de comunicación es necesaria para comunicarse con el CC1101. A todo lo anterior se suma que el modelo PIC32MXX75F256L cuenta con un controlador de red integrado que le facilitará la conexión a una red Ethernet (10 ó 100 Mbps), a través de un transductor LAN conectado directamente con un dispositivo MagJack que integra la conexión por con un puerto RJ45 y el acople de impedancias por medio de un transformador de pulsos.

El PIC32 tiene una arquitectura MIPS32 con un pipeline de 5 estaciones, su reloj de sistema puede llegar a los 80MHz. Cuenta con una unidad de multiplicación y división de un solo ciclo. Tiene una memoria de programa de 512kB y 128kB de memoria para datos e interfaces de programación y depuración JTAG.

Está fabricado en un encapsulado TQFP de 100 pines en montaje superficial con un tamaño 12 x 12mm. En la figura 5.19 se ilustra el diagrama de bloques del integrado, y se resaltan los módulos utilizados para este proyecto.

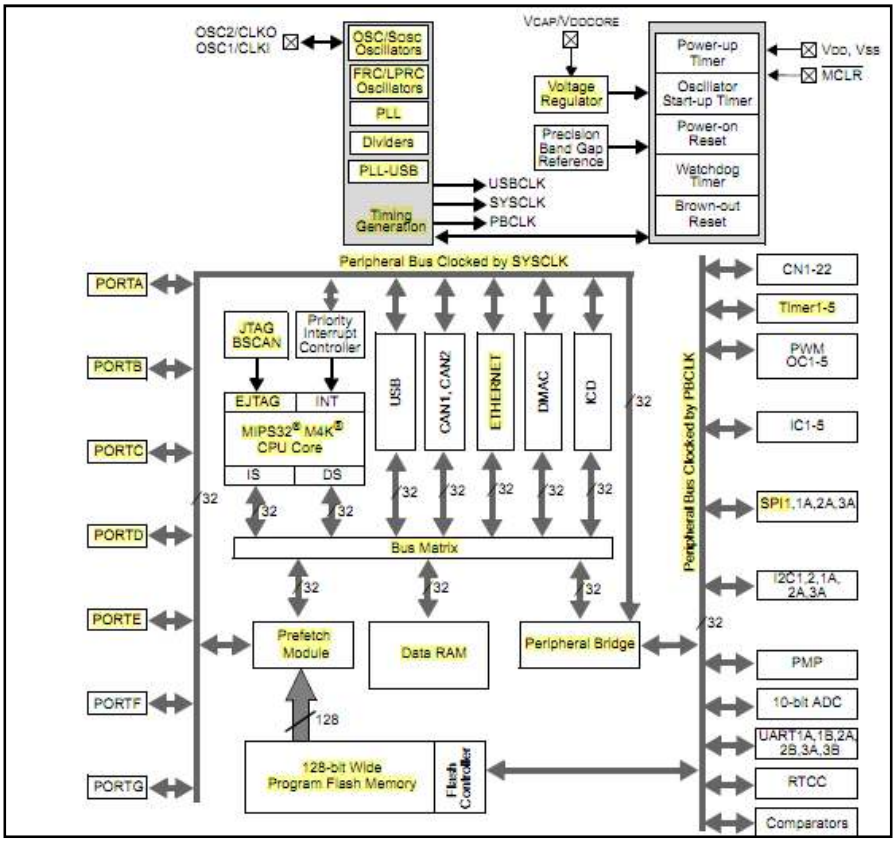


Figura 5.19 Diagrama de bloques del PIC32MX. Tomado de [41]

El controlador de interrupciones fue esencial para el manejo de todos los eventos generados por el transductor de radio y por el controlador integrado de Ethernet. Por otro lado los módulos temporizadores se usaron para ajustar tiempos de espera en el manejador del CC1101 (Timer1), para generar interrupciones de refrescamiento en la tabla de direcciones (Timer4&5) y para funciones de espera en general con propósitos de visualización (Timer3).

Los módulos Input Capture fueron necesitados para generar eventos de interrupción externos, dichos eventos vendrían del CC1101 por las salidas GDO0 y GDO2 y corresponden a las interrupciones de nivel umbral en la FIFO del transductor y final de paquetes. Si bien el PIC32MX cuenta con entradas para interrupciones externas, éstas no se pudieron utilizar ya que se encontraban ocupadas por otras funciones. En la tabla 5.8 se muestra que las patitas necesarias para la comunicación con el transductor LAN y las del

SPI1 utilizado como interfaz con el transductor RF, ocupaban todos los lugares para conectar interrupciones externas. Por suerte los módulos de captura de eventos pueden ser configurados para generar eventos de interrupción al bloque controlador de interrupciones.

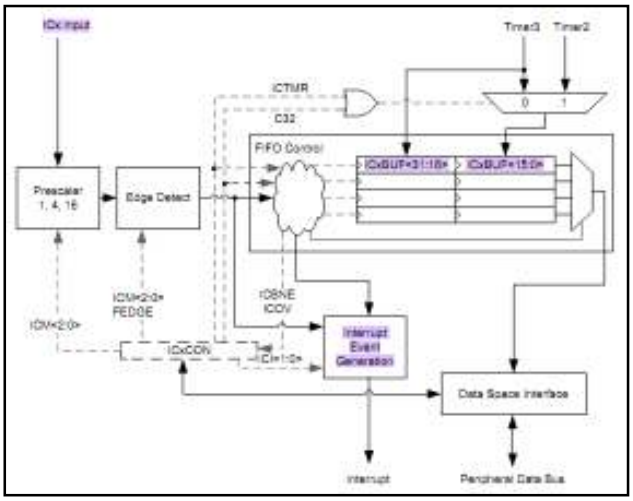


Figura 5.20 Diagrama de bloques del PIC32MX. Tomado de [42]

En la figura 5.20 aparece el diagrama funcional del evento de captura, como puede verse el evento de captura puede realizar conteos y almacenarlos en un valor compuesto por dos registros de 16 bits sin embargo, para los requerimientos del diseño únicamente era necesario la generación de un evento detectado como un flanco, positivo o negativo, en la entrada de ICx. Según la hoja de datos este módulo se puede configurar de esa forma y entonces no se utilizan los registros ICxBUF <31:0>, pero después de realizar varias pruebas se detectó una falla en la interrupción por una escritura no deseada en el buffer entonces, cada vez que se genera un flanco el buffer almacena un valor que no es leído, llenando así dicho buffer y generando un estancamiento que provoca interrupciones falsas. Para lidiar con esta situación fue necesario realizar una lectura dummy del buffer de conteos cada vez que se ingresa a la interrupción, al hacer esto el buffer nunca se llena y la interrupción se comporta de manera regular lo que permitió continuar con el diseño.

Los otros dos módulos clave para el desarrollo del Ethmesh fueron el controlador de comunicaciones para la interfaz periférica serie (SPI), y el controlador de red Ethernet.

5.2.3.2 Controlador de Ethernet integrado

El PIC32MX795F512L cuenta con un controlador de Ethernet integrado que a través de un chip de capa física (transductor LAN), implementa un sistema completo de nodo Ethernet. El controlador se compone de una serie de registros e interrupciones con un set de prioridades, dichas interrupciones se dirigen hacia el núcleo del PIC para ser atendidas y pueden provenir del controlador mismo o del chip de capa física. Otro componente importante son los DMA de transmisión y recepción que colocan los datos en la memoria en forma de bloques desordenados pero son direccionados por descriptores que mantienen su estatus y posición de memoria.

El controlador de acceso al medio maneja el acceso al canal y as pausas necesarias, así como el CRC y las operaciones de cálculo del checksum sobre las tramas Ethernet. Es fundamental un una interfaz de comunicación con el transductor LAN, para este caso se utilizó RMII. El RMII (Reduced MII) es una simplificación del MII (media independent interface), que utiliza solamente 8 de las 12 conexiones necesarias para MII y permite direccionar la conexión de la capa física a los switches Ethernet. En el anexo B.4 se presentan las conexiones necesarias entre el controlador integrado en el PIC32 y el chip de capa física.

5.2.3.3 El PIC32 Ethernet Starter Kit.

Este hardware integra todas las herramientas necesarias para la depuración y prueba de código, además cuenta con la interfaz de Ethernet necesaria para la implementación del Ethmesh. Para el desarrollo de este proyectos fueron adquiridos dos PICkits, uno de ellos con la interfaz de red y otro sin la interfaz de red que permitió realizar algunas pruebas de comunicación SPI.

Figura 5.21 Fotografía de los Kits adquiridos por el DIEC.

Como se muestra en la figura, el modelo de la izquierda cuenta con el conector para Ethernet, el PIC32MX795, el PIC32MX460 de depuración y control de la interfaz USB, la interfaz de programación mini USB, algunos LED´s e interruptores y el chip de capa física DP8384C.

El DP8384C realiza permite el acceso al medio, para ello realiza una auto negociación al momento que se conecta el dispositivo, en este proceso se establecen las características de la conexión, velocidad y estándar. Además se establecen posibles errores en el canal o problemas con los otros dispositivos conectados a la red. Este proceso se realiza cada vez que se conecta cualquier dispositivo a una red y tarda alrededor de 3 segundos. El transductor LAN toma los datos desde la interfaz RMII y les realiza un "scramble" para luego codificarlos en NRZI y los convierte en MLT-3, finalmente los pasa por un driver de transmisión que los envía hacia el transformador de pulsos que realiza el acople de las impedancias entre la tecnología TTL y la línea analógica.

Muchos de los bloques en la figura 5.11 están integrados en este kit de desarrollo, en la figura 5.22 se muestra el microcontrolador principal y sus conexiones con el resto del hardware en la tarjeta.

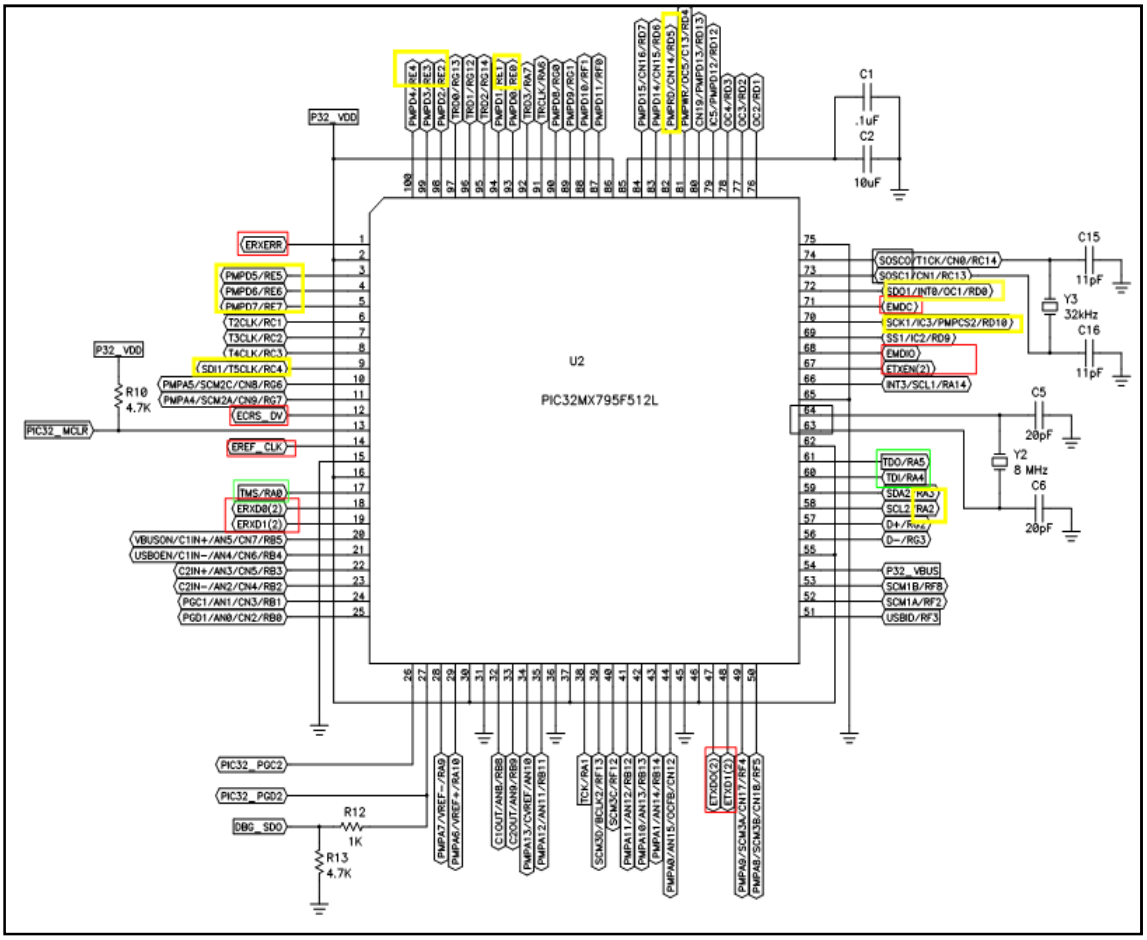


Figura 5.22 PIC32MX y sus conexiones con el resto del hardware.

En la figura aparecen sobresaltadas todas las conexiones que el micro tiene con el microcontrolador de depuración a través de JTAG (verde), las conexiones por medio de RMII hacia el DP8384C (rojo), los cristales externos para le frecuencia de operación del core (negro) y las salidas hacia un conector del fabricante Hirose (amarillo), que se utilizó para realizar el añadido de la etapa de radio.

5.2.4 Diseño de una expansión para la etapa de radio

Para proporcionar capacidad de conexión al PICKit fue necesario construir una placa con una etapa de radiofrecuencia, la cual fue conectada con el PIC32MX por medio de una interfaz de expansión, como se muestra en la figura 5.23 para dicha interfaz Microchip utiliza un conector del fabricante Hirose de 120 conexiones a señal y 12 contactos a tierra.

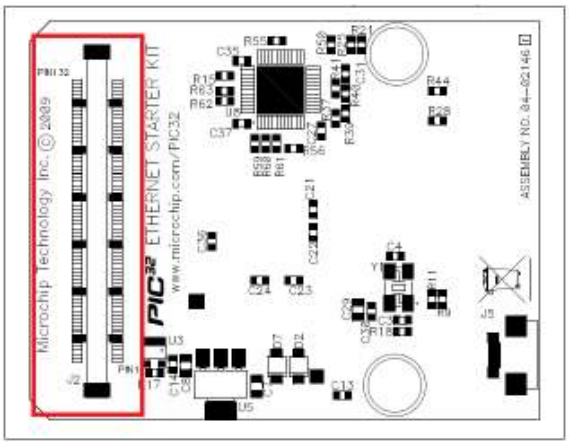


Figura 5.23 Bottom Layer del PICKit.

Por lo tanto fue necesario adquirir un receptor para este tipo de conectores, además para el diseño del PCB se hizo la edición del footprint para este conector, la figura del footprint final se muestra en la figura 5.24.

Figura 5.24 Diseño del footprint para el conector Irose.

Para el diseño de cualquier placa se deben considerar los filtros adecuados para eliminar ruidos entre las vías de comunicación. En la figura 5.25 se presentan los valores

agregados para la etapa de radio, además se muestran las conexiones que se extrajeron del conector Hirose hacia una terminal de 22 pines, en la terminal quedan a disposición los pines correspondientes a la parte baja del puerto E y algunas opciones de comunicación extra. Se dejó prevista una interfaz de comunicación UART para fines de prueba, sin embargo no se llegó a utilizar durante el desarrollo del proyecto, estas son solo previsiones que se dejan dispuestas para situaciones que pueden emerger, pero que no necesariamente se llegan a utilizar en un proceso de diseño.

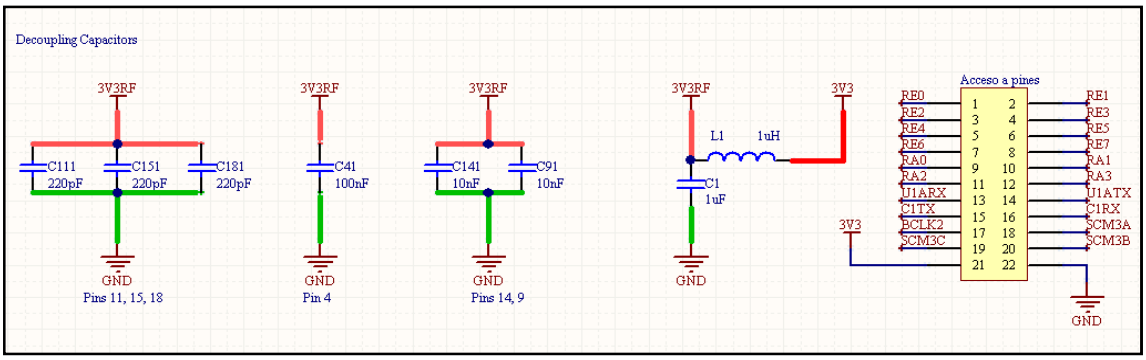


Figura 5.25 Etapa de desacople y filtrado de la placa de radio.

Como se muestra en la figura, una pequeña bobina es colocada entre las alimentaciones de la etapa digital y de radio para prevenir que posibles picos generados en la etapa de radio lleguen a la etapa digital y la dañen. Las conexiones llevadas hasta el CC1101 para establecer la comunicación y el manejo de los datos se presentan en la figura 5.26, allí aparece la representación en el esquemático realizado en Altium para el conector, con las conexiones necesarias y los pines conectados a tierra.

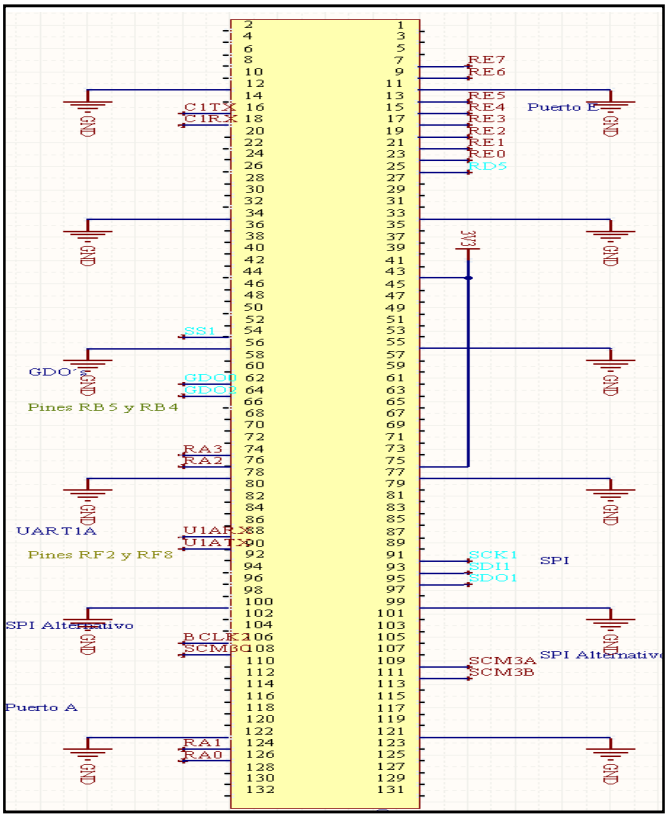


Figura 5.26 Esquemático y salidas del conector Hirose.

El resto del esquemático se presenta en la figura 5.27, en esta aparece el CC1101 con el circuito de desbalance para la salida. El dimensionamiento de los componentes se tomó de una nota de aplicación recomendada por el fabricante [2]. El valor exacto de los componentes del circuito balun es muy importante, ya que es la única garantía de que las comunicaciones entre el integrado y la antena se van a dar de la manera adecuada.

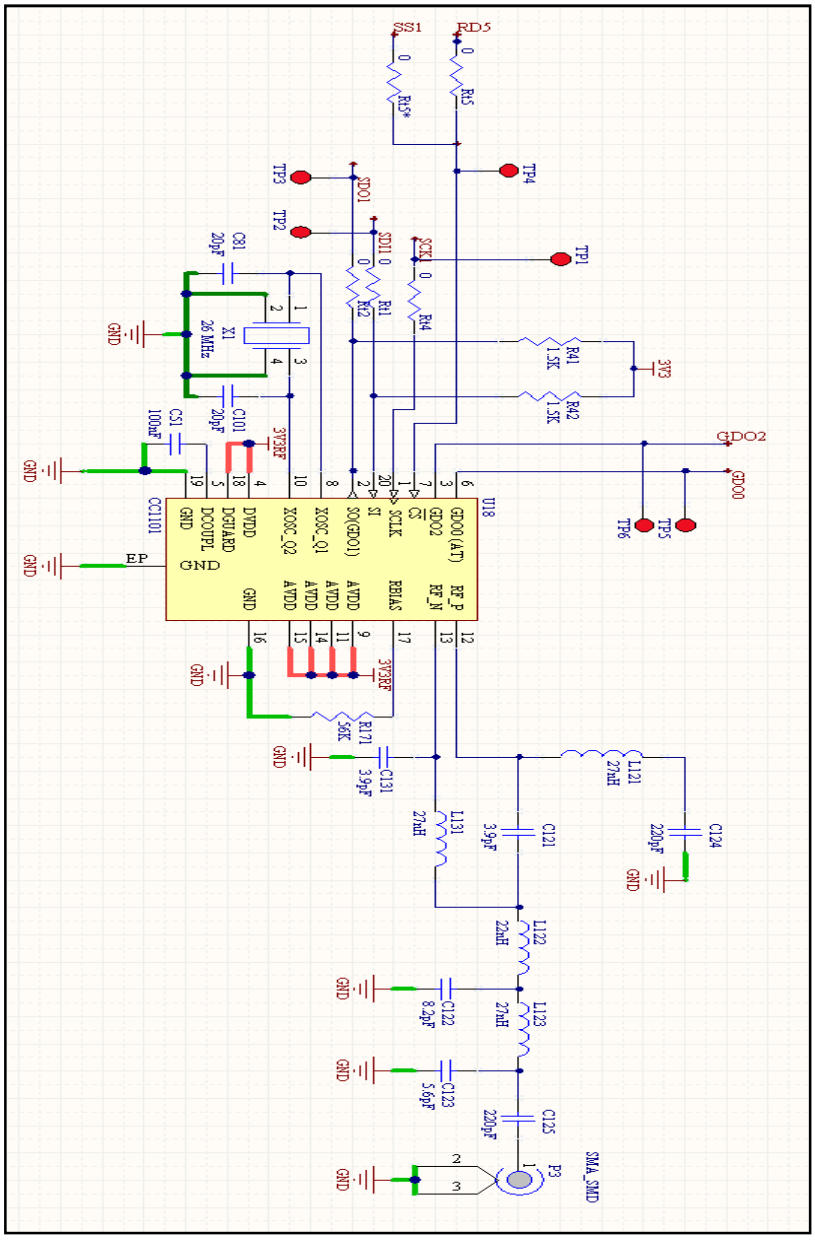


Figura 5.27 Esquemático del CC1101 y componentes para trabajar entre 351-433MHz.

El encapsulado del CC1101 tiene la característica especial de tener un área metalizada expuesta hacia la capa superior de la placa, dicha superficie debe ser aterrizada y se utiliza para el enfriamiento del integrado cuando éste se encuentra disipando potencia. Por ello fue necesario hacer algunas consideraciones en el desarrollo del PCB para la placa y a la hora de montar el dispositivo con el objetivo de proporcionar el mejor funcionamiento al CC1101.

Para realizar las pruebas de la mejor forma se insertaron el PCB espacios abiertos para entre las líneas del SPI, de forma que cuando se montara la placa con el Kit se pudieran medir las 4 líneas del SPI con un osciloscopio para corroborar que las conexiones salientes del conector Hirose estuvieran correctas. Una vez descartada la presencia de fallos se procede a cerrar los espacios con resistencias de 0Ω. En la figura 5.28 se observan círculos grises un poco más grandes que los demás, éstos corresponden a los puntos de prueba para las 4 líneas del SPi y las dos interrupciones GDOx que entran al PIC. A la izquierda pueden verse un grupo de 7 resistores con un footprint un poco más grande, estos resistores fueron los utilizados para cerrar las líneas tal y como se mencionó antes.

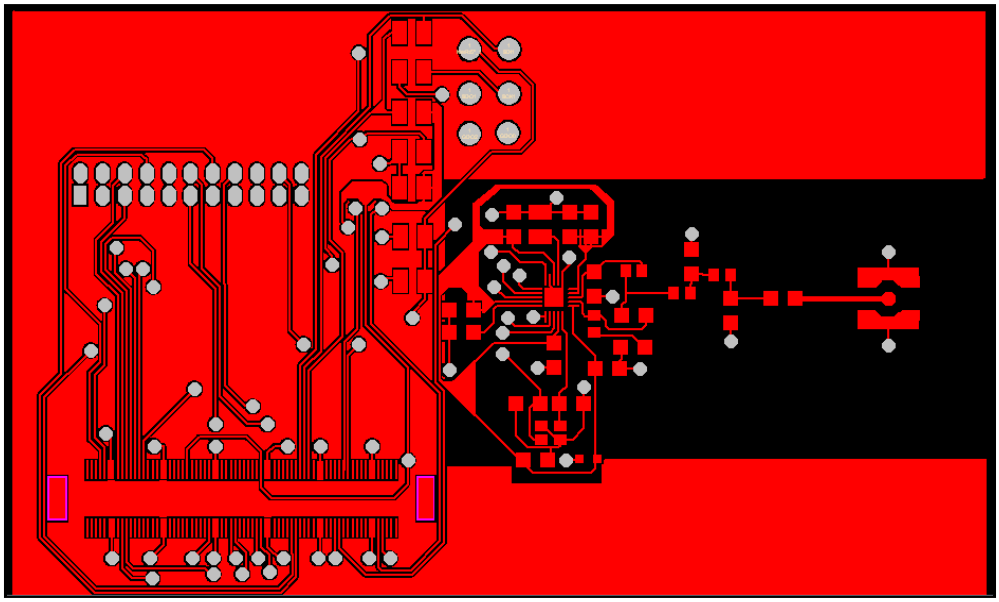


Figura 5.28 PCB fresado para el montaje de la etapa de radio (Top Layer).

Algunas otras consideraciones fueron evitar el trazado de líneas con márgenes agudos cerca de las líneas del cristal, ya que esto puede interferir en el cristal y producir ciclos de trabajo irregulares. También fue retirada una sección del plano de tierra cerca del conector SMA de para la antena, esto para que la antena monopolo que se colocó no comparta un plano horizontal con la tierra de la etapa digital. El conector debe tener un acople de 50Ω y facilita la rápida conexión o desconexión de la antena.

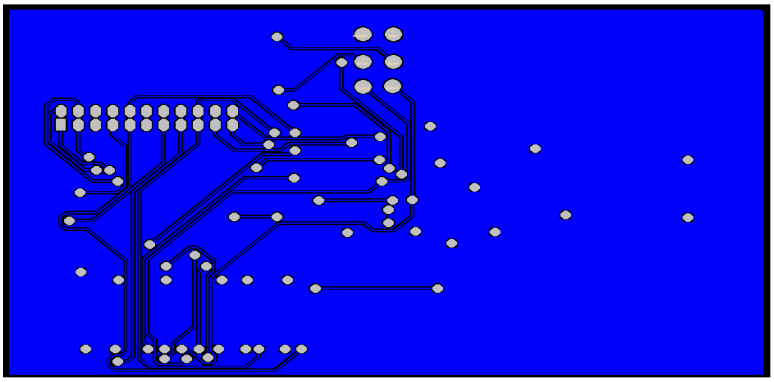


Figura 5.29 PCB fresado para el montaje de la etapa de radio (Bottom Layer).

El top layer debe ser utilizado para el trazado del as rutas para las señales y los espacios no utilizados deben ser metalizados y conectados a tierra usando varias vías. Como ya se ha mencionado el filtrado es importante, deben colocarse capacitores entre los puntos de alimentación y tierra. Por último los componentes usado fueron lo más pequeños posible (0805,1210), y de montaje superficial.

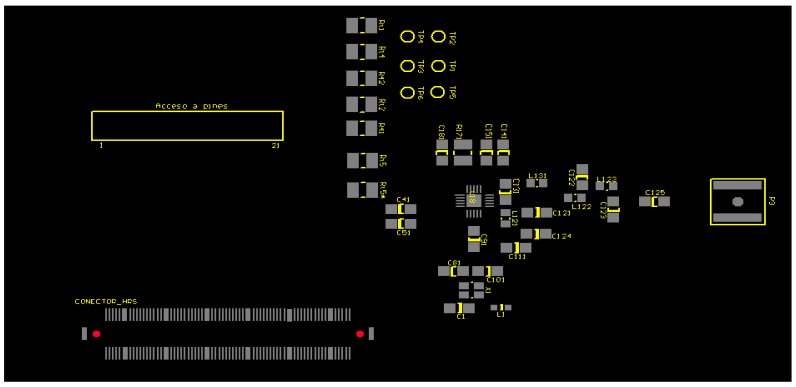


Figura 5.30 PCB fresado para el montaje de la etapa de radio (Top paste).

5.3 Descripción del software

El software desarrollado a lo largo de este proyecto puede dividirse en dos secciones principales, el software para el Ethmesh y el software que reside en la PC. Ambos se complementaron durante todo el desarrollo de este diseño ya que sin uno o el otro no hubiera sido posible completar los objetivos planteados.

El software para la PC comenzó a desarrollarse con la idea de utilizarlo para hacer pruebas sin embargo, con forme las funciones tanto del Ethmesh como las ofrecidas al usuario en la PC fueron aumentando, este software de prueba se fue convirtiendo en una pieza fundamental del proyecto. Pero por otro lado, siempre se mantuvo todo lo relativo al funcionamiento del Ethmesh independiente del software para la PC, lo anterior quiere decir que el dispositivo construido está diseñado para funcionar con cualquier otro software mientras cumpla con algunas condiciones, dichas condiciones serán reveladas a lo largo de las siguientes secciones. Esta flexibilidad le permitirá a cualquier programador desarrollar su propio software para comunicarse con el Ethmesh, dejando abierta la posibilidad de crear un software más amigable para el usuario final, en caso que así se deseara.

5.2.4 Diseño del software para el Ethmesh.

El software para el Ethmesh reside en el microcontrolador PIC que se describió en secciones pasadas, aunque existen dos dispositivos fundamentales que son externos y cuentan con su propio firmware (el DP8384C y el CC1101), no se realizó ninguna programación en ellos. La herramienta de desarrollo de software para PIC empleada fue MPLab. La razón para utilizarlo fue que ya se contaba con algunas bases en el uso de este ambiente de desarrollo, además el software viene incluido con los Kits de desarrollo adquiridos. MPLAB cuenta con herramientas de visualización de registros, simulación y depuración que permite colocar puntos de parada en el código; estas utilidades fueron clave a la hora de realizar pruebas y tomar resultados.

5.2.4.1 Manejo del Stack TCP/IP de Microchip

El Stack de TCP/IP se adquiere directamente de la página de microchip, allí mismo existen algunos links que proporcionan ejemplos de aplicación y otra documentación. Lo que se descarga es un ejecutable que instala en la raíz un folder con todos los archivos necesarios, estos archivos serán utilizados por el compilador C32 asociado al MPLAB, entre ellos se encuentran todos los header y los archivos punto c que conforman el Stack. Para esta parte del proyecto fue necesario realizar un estudio de dichos archivos, a fin de determinar cuáles eran necesarios de incorporar en el programa para el Ethmesh, la manera de hacer esto es entendiendo que el Stack permite hacer uso de las comunicaciones sobre una red Ethernet a partir de distintas capas, en este caso se utilizó el acceso desde la última capa posible, los sockets BSD (véase 3.6).

El Stack está compuesto de un gran número de archivos fuente, dentro de cada uno de ellos residen las funciones que se pueden llamar para realizar las comunicaciones, estas funciones dependen de otras a su vez. En el esquema de la figura 5.31 se presenta un diagrama de las funciones relacionadas al Stack TCP/IP que son invocadas durante la inicialización del sistema y dentro del lazo infinito.

Los TCP_TICKS_PER_SECOND son una unidad para la temporización de varios eventos relacionados con la recepción, la transmisión y el almacenaje de los datos. Como puede verse en la figura 5.31, los ticks se inicializan con una función del archivo "system_services" que recibe el reloj del sistema, 80MHz en este caso, y una constante definida con un valor de 100, con este valor se ajustan eventos de TICK cada 20ms. Los ticks son generados utilizando una interrupción del núcleo (CoreTimerInt), y se configura para generar una interrupción del núcleo cada 10ms, en cada una de estas interrupciones se desplazan otras cuentas que son conocidas como los tcpticks. Los TCPTICKS funcionan como base de tiempo para ajustar las definir los periodos de espera en retransmisiones TCP, purgas de los buffer de los sockets, solicitudes ARP, tiempos de espera por colisión, entre otros.

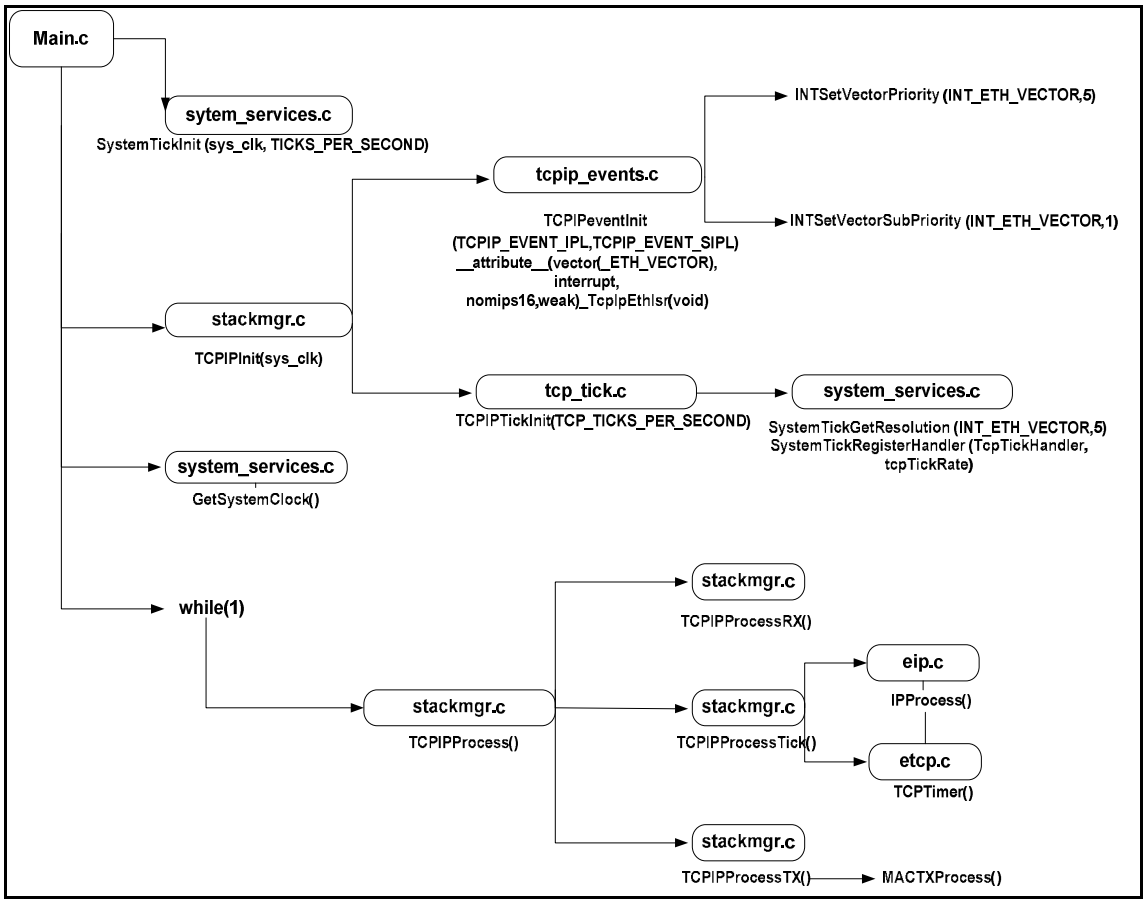


Figura 5.31 Esquema de funciones del Stack utilizadas.

Además de temporizaciones el Stack utiliza interrupciones del controlador Ethernet integrado en el PIC, en la figura 5.31 puede verse que la definición para la prioridad de estos eventos se establece mediante funciones del archivo *tcpip_event.c*. Al vector de interrupción del controlador de red se le asignó una prioridad de 5 y con una sub-prioridad de 1, esto es importante de anotar ya que para el resto de las interrupciones será necesario conocer los valores de las prioridades que se van asignando. Estos valores de prioridad para los eventos del controlador se pueden cambiar desde los archivos *tcp_bsd_config.h.* En *hardware_profile.h* se pueden configurar las prioridades del temporizador del núcleo, el temporizador del núcleo es una manera alternativa de generar interrupciones cada cierto tiempo de manera más precisa ya que no requiere del uso directo de temporizadores. Para acceder a las configuraciones del núcleo es necesario cambiar las palabras de configuración del microcontrolador, se han asignado prioridad 4 y sub prioridad 1 al core timer. Esto quiere decir que las interrupciones de algún evento TCP tienen mayor prioridad

que los conteos del núcleo, esto garantiza que los eventos de Ethernet no sean interrumpidos.

Accediendo a las palabras de configuración también es posible configurar los divisores y multiplicadores de frecuencia a la entrada del cristal oscilador, para obtener cualquier frecuencia de sistema deseada, en este caso se ajustó la más rápida posible que corresponde a 80MHz.

En conclusión lo que debe tenerse en cuenta para el uso del Stack es el orden las interrupciones generadas por el núcleo y por el controlador de Ethernet a fin de no utilizar vectores de interrupción en el PIC que coincidan son los que ya han sido utilizados, además tener en claro las prioridades asignadas de manera que se tenga conocimiento de cuáles eventos podrían eventualmente tener prioridad sobre las interrupciones que se agreguen para el manejo de las comunicaciones por radio, sin embargo la probabilidad de que ocurran ambos eventos es baja debido a la diferencia de velocidades entre la comunicación de Ethernet y la comunicación de RF.

El otro detalle importante es el uso de los sockets para las comunicaciones UDP, para transmitir un datagrama es necesario configurar un socket de tipo UDP. Un socket está compuesto por una dirección IP y un número de puerto que indica el tipo de servicio. En esta caso se tienen tres puertos y por ende es necesario crear tres sockets y realizar una revisión continua de los mismos, el sistema de ticks del Stack permite colocar un socket en estado de escucha y revisarlo después de haber pasado el control a otras tareas, esto se conoce como un sistema multitarea.

Para la creación de los sockets se utilizan funciones muy similares a las del punto 3.6.2. En el siguiente diagrama se muestra la lógica para la creación y uso de un socket BSD, como puede verse en la figura es cuestión de declarar la estructura descriptora y crear el socket bajo algún identificador, luego asociarlo a una estructura de dirección que contiene una dirección IP cualquiera para recibir, el número de puerto y el tipo de mensaje

(datagrama o stream). Luego se atienden las solicitudes de el puerto y se responde, hay que recordar que la función de recibir carga una IP al descriptor del socket y por ende, se estará regresando el mensaje a la misma IP que recién fue atendida.

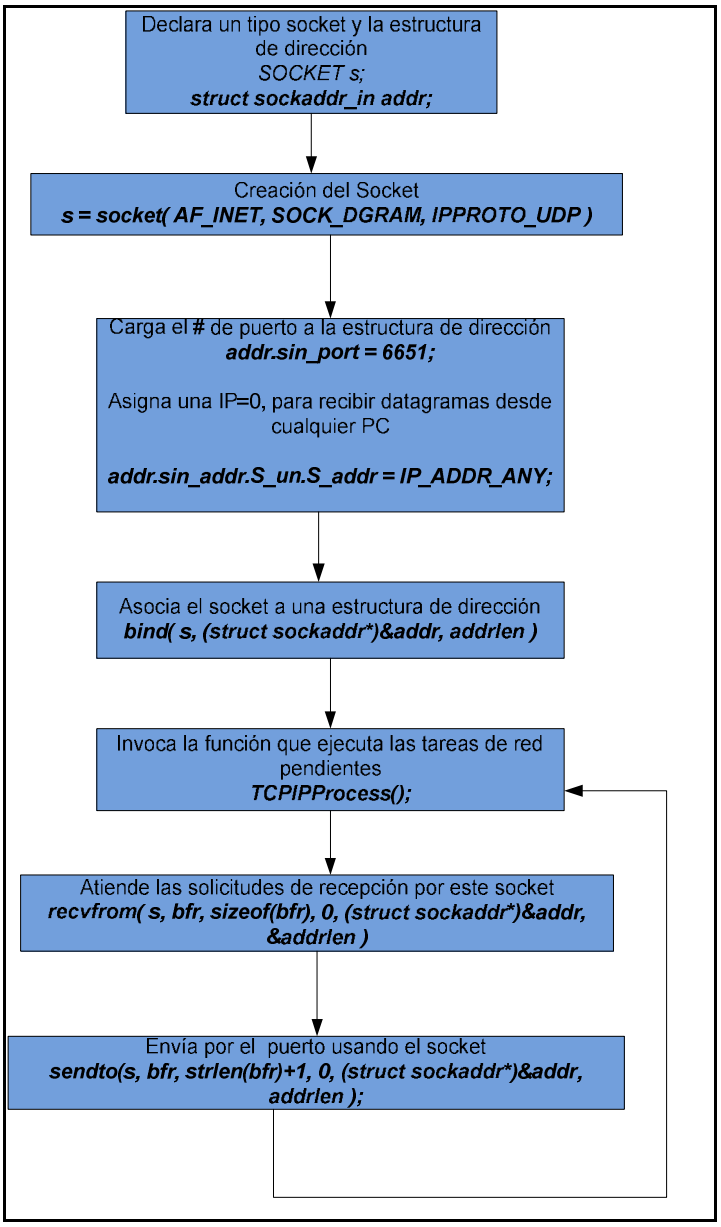


Figura 5.32 Diagrama de flujo para creación y uso de un datagrama.

En la figura 5.32 puede notarse que existe un lazo de regreso a la ejecución de la función TCPProcess(), ésta es la función que ejecuta la máquina de estados de recepción y los procesos del Stack, además recorre las solicitudes en el ARP FIFO y atiende los

paquetes con ARP resuelto descartando aquellos paquetes con tiempo de ARP expirado. También retransmite mensajes para los que se excedió el tiempo de reconocimiento.

5.2.4.2 Diseño de un control para el CC1101

Una vez analizado el comportamiento y las configuraciones requeridas para el CC1101 se procedió a programar un control para el CC1101, lo primero fue escribir las funciones de configuración mediante subrutinas de acceso a los registros del CC1101 por medio de la interfaz SPI.

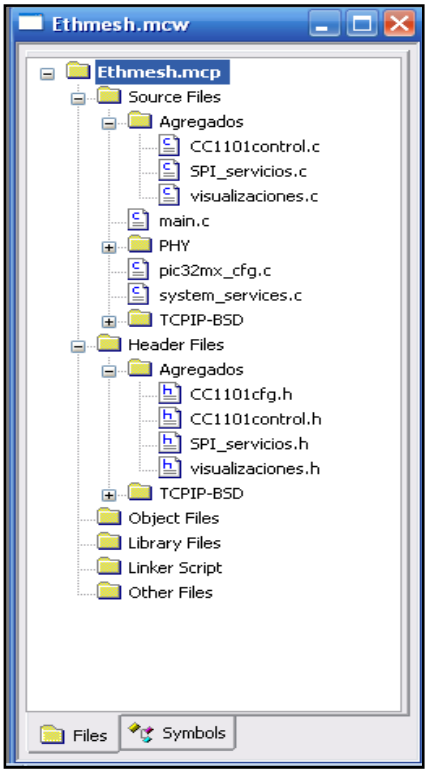


Figura 5.33 Árbol de archivos para el proyecto Ethmesh.

En la figura 5.33 se muestran desplegados los archivos que forman parte del software del Ethmesh. Las carpetas TCPIP_BSD contienen todas las funciones y encabezados de las capas 3 y 4 así como las funciones propias de los sockets, en "PHY" se encuentran los manejadores del MAC a través del DP8384C, que corresponden a las capas 1 y 2.

En la carpeta de “Agregados” de los archivos fuente se muestran los tres archivos agregados al Stack. Por otra parte se pueden ver los archivos encabezados en la parte inferior de la figura, estos encabezados contienen los prototipos de las funciones escritas y algunas definiciones útiles. En “SPI_servicios.c” se encuentran muchas de las subrutinas para la inicialización del SPI, configuraciones de registros del CC1101, envío de comandos o strobes, carga de las configuraciones por defecto, lectura de un registro de configuración y lectura burst de varios registros consecutivos. A su vez, “SPI_servicios..c” utiliza las definiciones del “CC1101cfg.h” y de “CC1101control.h”. En la tabla 5.9 se exponen las subrutinas y funciones contenidas en “SPI_servicios”, éstas corresponden a los servicios más básicos que se pueden realizar sobre el CC1101.

Tabla 5.9 Funciones contenidas en SPI_servicios.c

Función o subrutina	Entradas	Salidas	Descripción
Select y Unselect	-	-	Bajan o suben el Chip Select con una pequeña espera antes de subir.
ConfigTimer1	-	-	Configura el timer 1
SpiInit	-	-	Configura el SPI1 con una velocidad de 1MHz para SCLK
cfg_write	**unsigned char** Reg **char** valor	Retorna el byte de estatus del CC1101	Recibe un registro y el valor a cargarle
strobe	**unsigned char** Strobe **unsigned int** idle **char** CSn	Retorna el byte de estatus del CC1101	Recibe un comando y un tiempo de espera luego de ejecutarlo. Usada para mover de estado al transductor.
default_config_CC110	**-**	-	Configura todos los registros del CC1101 con valores fijos.
cfg_read	**unsigned char** Reg	Retorna el byte de estatus del CC1101	Lee un registro de configuración solicitado.
cfg_b_read	**unsigned char** Reg_In **unsigned char** Reg_Fin **char** Cnfg[]	Carga todos los registro leídos en un arreglo Cnfg[].	lectura burst para registros de configuración contínuos, si maneja el CSn

Otros servicios relacionados con el CC1101 que son un poco más complejos se encuentran dispersos entre los archivos CC1101control.c y visualizaciones.c, de igual forma que en el caso anterior, se presentan en una tabla las funciones que tienen que ver con el manejo del CC1101 y se encuentran contenidas en dichos archivos.

Tabla 5.10 Funciones relacionadas los accesos contenidas en CC1101.c y visulizaciones.c

Función o subrutina	Entradas	Salidas	Descripción
status_read	**unsigned char** Stat	Valor del registro solicitado	Rutina de lectura de registros de status
FIFO_write	**int** Bytes_to_Write **int** txBufferIndex BYTE TxBuffer[]	Byte de estatus	Rutina de escritura a la FIFOTX del CC1101, recibe el buffer desde el que se va a escribir y un offset.
Read_FIFO_RX	**int** Bytes_to_Read **int** pBufferIndex BYTE rxBuffer[]	-	Rutina de lectura desde la FIFORX del CC1101
CC1101Initialization	-	Devuelve falso si el CC1101 se niega a entrar en espera para ser configurado	Rutina de inicialización del CC1101, utiliza una pequeña máquina de estados que envía a reset, espera, lleva a idle, comprueba que este en espera y configura.
Config_GDO	BYTE GDO_X BYTE Config	-	Configura la funcion de uno de los dos GDO. Recibe definiciones que están relacionadas al tipo de interrupción generada.
Load_config	**char** Cnfg[]	Escribe en el buffer Cnfg[] todos los bytes leídos	Mueve el control del CC1101 al estado de Idle y lee todos los registros de configuración.
Config_Pkt_Size	**char** Bfr[]	Retorna un confirma de fallida o realizada en el buffer Bfr[]	Configura la tamaño de los paquetes

Tabla 5.10 Funciones relacionadas los accesos contenidas en CC1101.c y visulizaciones.c (continuación)

Función o subrutina	Entradas	Salidas	Descripción
Report_Status	**char** Bfr[]	Reescribe el valor leído en la posición cero de Bfr[] y coloca un 0x00 en Bfr[1]	Lee un registro de estatus
Config_All	BYTE Cfgs[] **int** cantidad	escribe el contenido de Cfgs[] y lo carga en los registros de configuración del CC1101	La cantidad de direcciones de registro + valores a agregar está dada por "cantidad"

De las funciones anteriores es importante resaltar la de inicialización del CC1101, ya que una correcta inicialización fue clave para el buen funcionamiento del integrado. En la figura 5.34 se presenta el diagrama de flujo de la pequeña máquina de estados programada en el PIC.

El uso de todas las funciones anteriores se facilita usando definiciones que se encuentran en los archivos .h, de estos archivos el que contiene mayor cantidad de definiciones es CC1101cfg.h. En éste se encuentran todas las constantes utilizadas para los tiempos de espera, direcciones de los registros, comandos, direcciones de los registros de status y de las memorias FIFO de transmisión y recepción. En el anexo B.5 se presenta un mapa completo de los registros de configuración, comandos, estatus, y acceso a las memorias, además aparece en la parte superior de la tabla el offset necesario según el tipo de acceso, por ejemplo para escribir el registro IOCFG2 de manera sencilla se usa la dirección 0x00, si se desea hacer por escritura burst se debe leer como 0x40. De forma similar, si se lee 0x80 entonces es de manera simple pero con 0xC0 se accede de manera burst. Las FIFO_RX y FIFO_TX únicamente son accedidas por burst, para los strobes es indiferente el valor en el bit de lectura-escritura.

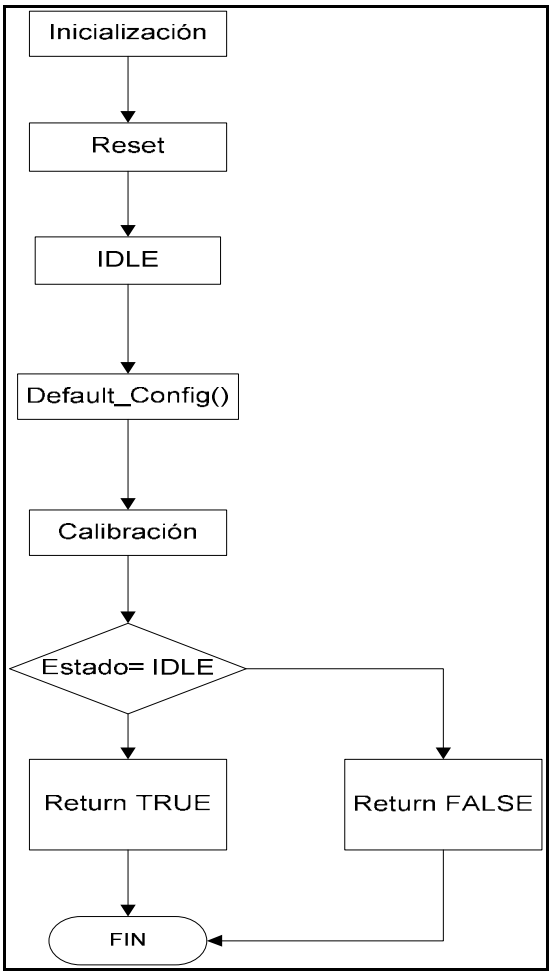


Figura 5.34 Inicialización del CC1101.

Ahora que se ha explicado la manera en que se lleva a cabo la inicialización y la manipulación de los registros y los estados del CC1101 resta sólo una cosa: plantear cómo se hizo la recepción y transmisión de paquetes en forma ordenada con el resto del software. El concepto es sencillo, se hizo uso de las interrupciones pero, la idea es relatar en qué forma. Como ya se ha discutido en secciones anteriores, el PIC32MX cuenta con un manejador de interrupciones, éste es capaz de manejar 7 niveles de prioridad de interrupción y 4 niveles de sub prioridad, el stack utiliza las prioridades 4 y 5, dejando tres interrupciones más por debajo. Se decidió utilizar únicamente las prioridades por debajo de las generadas por el controlador Ethernet, para darle mayor importancia a la recolección de mensajes y solicitudes UDP. Esto por una sencilla razón, la diferencia entre las velocidades de ambas interfaces. Es obvio que la Ethernet es mucho más rápida que un canal inalámbrico de 38,4kbaud, por eso se le da mayor prioridad a la comunicaciones por red,

de manera que en caso de darse dos eventos simultáneos, se atienda primero a aquel que es más rápido, de manera que el otro no tenga que esperar mucho y así ayudar a reducir el cuello de botella que se forma producto de la gran diferencia en las velocidades. La tabla 5.11 resume los vectores de interrupción utilizados en el software del Ethmesh.

Tabla 5.11 Vectores de interrupción utilizados para el software Ethmesh

Evento	Vector	Prioridad	Sub prioridad
Evento de TCP/IP	48	5	1
Core timer	0	4	1
InputCapture2 (GDO2) Fin paquete	9	3	3
InputCapture5 (GDO0) Threshold	21	2	3
Timer4&5 (TTL refresh)	20	1	1

En el diagrama de tiempos de la figura 5.35 se muestra la lógica en la generación de las interrupciones, los pines de control GDO0 y GDO2 permiten controlar los eventos relacionados a las transmisiones y las recepciones, de forma que el GDO0 ha sido programado para notificar cuando se sobrepasa o se vacía por debajo de umbral y el GDO2 es el fin de paquete. El funcionamiento de la interrupción de fin de paquete es sencillo: la patilla GDO2 se levanta cuando se ha terminado de enviar (en TX) o recibir (en RX) una palabra de sincronización y se devuelve a cero cuando se ha terminado de recibir o transmitir un paquete. Esto se lleva a cabo mediante un contador interno que controla el fin de la operación mediante el valor pre cargado en un registro que define el tamaño de paquete.

Para el análisis de GDO0 es conveniente considerar los casos de transmisión y de recepción por separado. En una transmisión permite conocer el momento en que se dispone de espacio en la FIFO_TX para colocar los bytes que se van a enviar. El proceso de escritura procede así: una vez que se comienzan a escribir los datos y se llega a un

determinado nivel de llenado en la FIFO_TX, 33 bytes en este caso, se levanta la patilla de GDO2, el micro continúa la escritura de una cantidad de bytes no mayor a la capacidad de la FIFOT_TX. Después de haber escrito el paquete, es necesario poner al traductor en estado de transmisión, después la señal GDO2 bajará de nuevo en el momento en que se hayan transmitido una cantidad de bytes tal que, la FIFO esté por debajo del nivel de umbral (33bytes). En ese momento es posible reescribir una cantidad máxima de bytes equivalente al nivel de umbral menos uno (32bytes), cada recarga se realiza entonces cuando el GDO2 lo indique mediante un flanco negativo. Y el fin de paquete, GDO0 indica el momento en que se ha terminado el envío de un paquete.

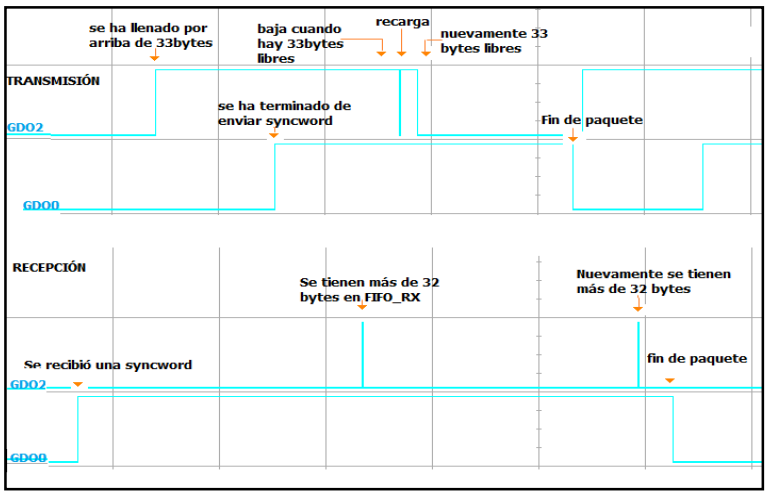


Figura 5.35 Comportamiento de GDO0 y GDO2 en TX y RX.

Es necesario considerar por aparte los casos en que un paquete es mayor o menor que los 64bytes disponibles en la FIFO_TX. De manera que para una transmisión de tamaño menor a 64bytes, bastará con una sola escritura a la memoria del CC1101 y por ende, no serán necesarias las recargas controladas por el nivel de umbral (GDO2) y por ello la interrupción generada por dicha señal debe ser ignorada en estos casos. Ahora que si se tratara de una transmisión de un paquete mayor a 64bytes, entonces es necesario llevar un control de cuántas recargas son necesarias y sí es esencial el uso de la señal GDO2 para el control de las recargas. Para ambos tamaños de paquetes la señal GDO0 indicará el final de una operación de transmisión para un paquete.

En la parte inferior de la imagen de la figura 5.35 se puede observar el comportamiento de las GDOx en una recepción. En este caso la señal GDO0 se levanta en el momento en que se va a comenzar a recibir un paquete sin embargo, esto no genera un evento al micro ya que no es un momento para tomar ninguna acción. Lo que sí es importante es evitar que la FIFO_RX se desborde, por ello se toma el flanco positivo de GDO2 para generar un evento de lectura de 31 bytes desde la FIFO_RX. De este modo quedan disponibles para que continúen llegando los bytes hasta que GDO0 indica el fin del paquete y se pueden leer los bytes que hayan quedado restantes. Llevar el control de la cantidad de bytes restantes es responsabilidad directa del microcontrolador, para ello se incorporan en el código variables de control. En los siguientes diagramas se ilustra con mayor detalle las operaciones realizadas en el software para cada interrupción y otros detalles.

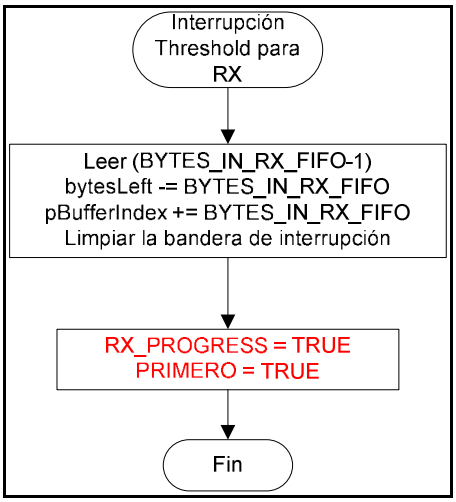


Figura 5.36 Interrupción de umbral para recepción.

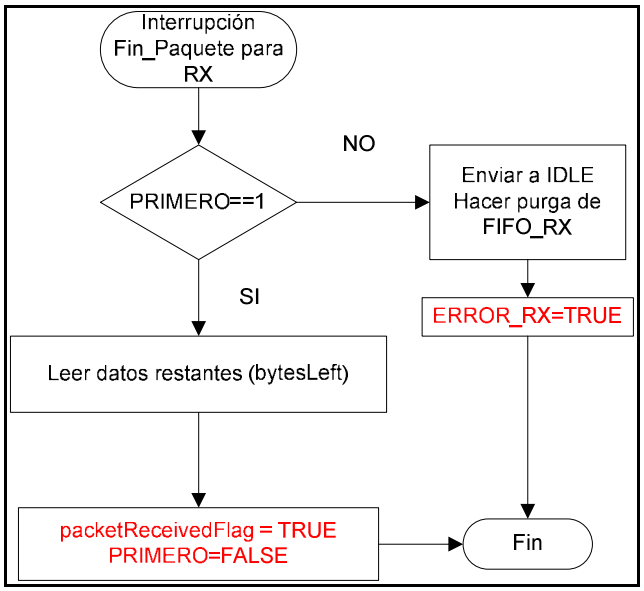


Figura 5.37 Interrupción fin de paquete para recepción.

Para el caso de una recepción, es necesario que en la interrupción de umbral se vaya actualizando la cantidad de bytes por leer en el paquete y un índice de que indique la posición en la que se deben cargar dichos bytes en un arreglo. El software del Ethmesh cuenta con un arreglo de 256 bytes "rxBuffer [256]", en el que se almacenan las fracciones de paquete que se leen en cada interrupción. De manera que por cada lectura se restan los bytes leídos al tamaño total del paquete (bytesLeft), así cuando llega el fin de paquete se tiene conocimiento de cuántos bytes quedan restantes para leerlos desde la FIFO_RX, evitando leer más bytes de la cuenta. Es importante notar que este método para manejar la recepción de paquetes funciona para cualquier tamaño de paquete, lo único que se debe hacer es pre configurar el valor de la variable bytesLeft que se usa para calcular el sobrante. En color rojo aparecen las variables de control de flujo, que son usadas para que la máquina de estados principal de Ethmesh se oriente a la hora de tomar decisiones, ambas variables son booleanas; una de ellas establece el momento en que se ha terminado de cargar un paquete completo al "rxBuffer".

La otra variable, llamada "PRIMERO", funciona para la corrección de una anomalía en el funcionamiento de la señal de fin de paquete. El problema que se detectó fue que la señal de fin de paquete se activa y desactiva cuando no se ha levantado la

interrupción de umbral, entonces se produce un descontrol ya que el PIC está programado para realizar una lectura de los bytes restantes en el flanco negativo de GDO2, entonces cuando luego se lee otra cantidad de bytes en el flanco positivo de GDO0 entonces se produce un exceso en las lecturas que desordena el intercambio de datos entre el PIC y el CC1101. El fenómeno se describe en la figura 5.38, compárese con el diagrama en la figura 5.35 y nótese la diferencia.

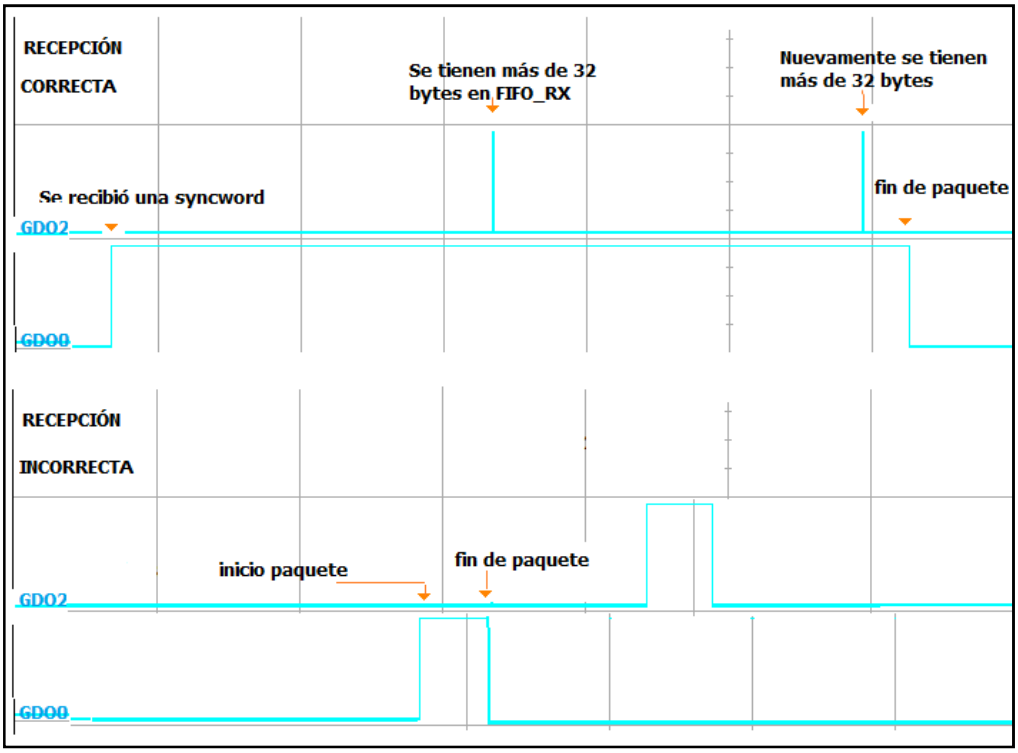


Figura 5.38 Interrupción incorrecta en GDO0.

De manera que la bandera PRIMERO detecta si se da una lectura previa en un umbral, de modo que en la interrupción de final de paquete se realiza un chequeo de la bandera, si todo anduvo bien el valor de PRIMERO debe ser verdadero, pero si es falsa para cuándo llega el fin de paquete quiere decir que no se ha presentado una condición de error. De esta manera se detecta el error, para corregirlo se hace una reprogramación de las señales GDOx y se purga la FIFO_RX.

Las transmisiones son acciones voluntarias por parte del control del Ethmesh, a diferencia de las recepciones que son eventos que dependen de la decisión de los otros nodos y no se tiene conocimiento del momento en que van a ocurrir, si se hiciera un polling de las recepciones por RF, éste debería ser siempre ejecutado y no sería posible atender a otras acciones.

Aunque las acciones de transmisión son acciones voluntarias es necesario usar interrupciones para realizarlas de la forma más eficiente. No es conveniente ocupar al núcleo del microcontrolador para estar pendiente durante todo el tiempo que tarde la transmisión, no sería eficiente ya que el tiempo que tarda el CC1101 para transmitir a 38,4kBaud es muy considerable en comparación con la velocidad de procesamiento del PIC y por ende, se podría aprovechar ese tiempo para realizar otras tareas. Para lograrlo la dinámica consiste en iniciar una transmisión y dejar que el CC1101 notifique al micro cada vez que necesite asistencia, esta asistencia puede ser una recarga de datos o una notificación de fin de paquete. El fin de un paquete es más una bandera para el control de flujo de la máquina de estados principal del Ethmesh, con esta bandera el control de flujo puede saber el momento en que cuenta con un paquete y además funciona para evitar que el PIC trate de transmitir un paquete en el momento en que el CC1101 se encuentre ocupado.

En el diagrama de flujo de la figura 5.39 se muestra el inicio de una transmisión, ésta es una acción previa a las interrupciones, básicamente consiste en cargar algunos datos sobre la memoria FIFO_TX y enviar al estado de transmisión. Primero se considera el tamaño del paquete a escribir en FIFO_TX, de ser posible toda la transmisión en una sola carga al CC1101 se deshabilita la interrupción de threshold. La bandera de control de flujo TX_PROGRESS se acierta para hacerle saber al control del micro que el transductor se encuentra realizando una transmisión, como se verá adelante esta bandera desencadena acciones sobre la memoria interna que el Ethmesh utiliza como colchón para lidiar con los cuellos de botella.

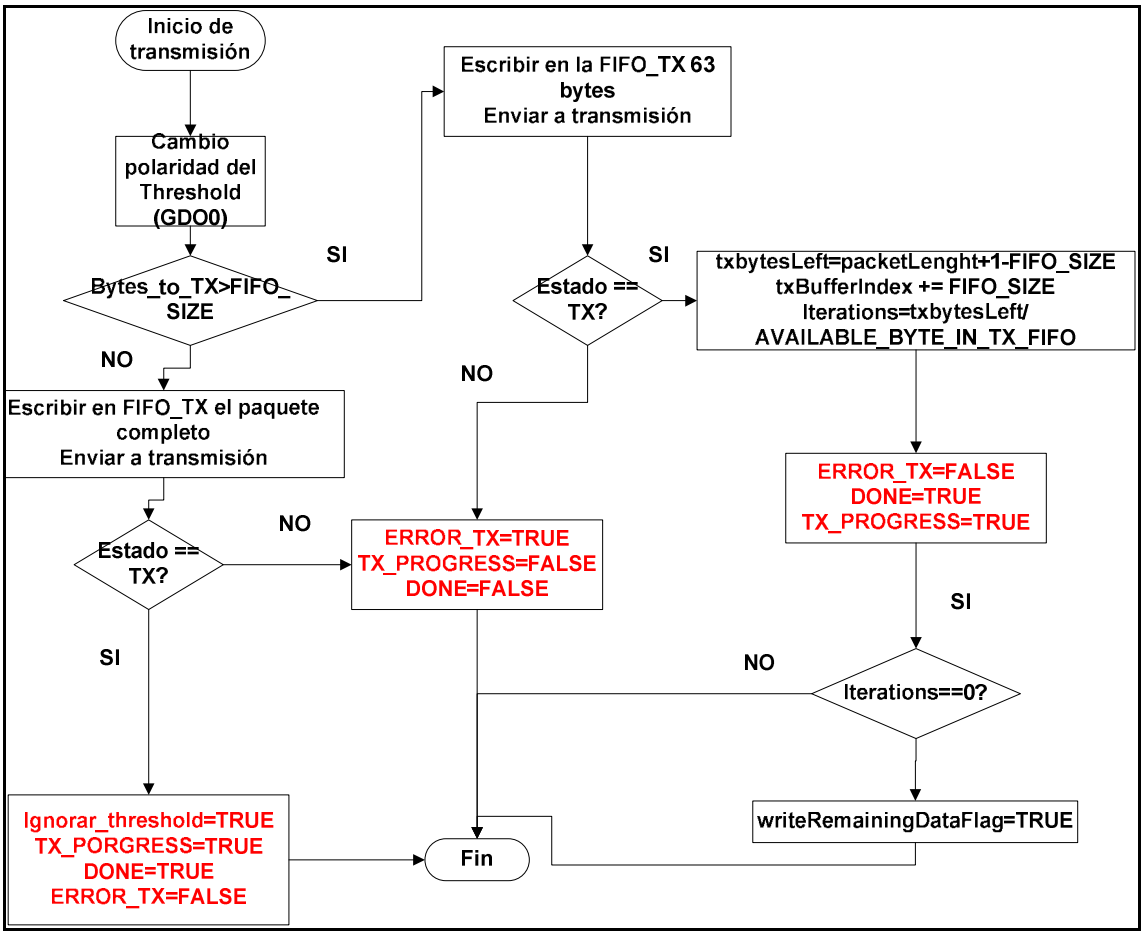


Figura 5.39 Inicio de una transmisión inalámbrica.

En caso de que el paquete no sea de menor tamaño que la capacidad de la memoria FIFO_TX entonces se debe hacer el envío de una parte del paquete, la variable "txbytesLeft" lleva el control de los bytes que faltan para completar el paquete, inicialmente se actualiza restándole los 63 bytes cargados, luego en las interrupciones de recarga se va disminuyendo por una cantidad equivalente al umbral de la FIFO_TX. El txBufferIndex es un índice para el buffer "txBuffer" de 256bytes, cada vez que se realiza una carga el índice se actualiza para mantener la posición desde la cual se deben ir haciendo las recargas. Por último la variable "iterations" es el cociente de los bytes restantes entre la cantidad que se irá recargando en cada interrupción de umbral, esta permite determinar cuándo se deben escribir los bytes restantes que no completan los 33bytes.

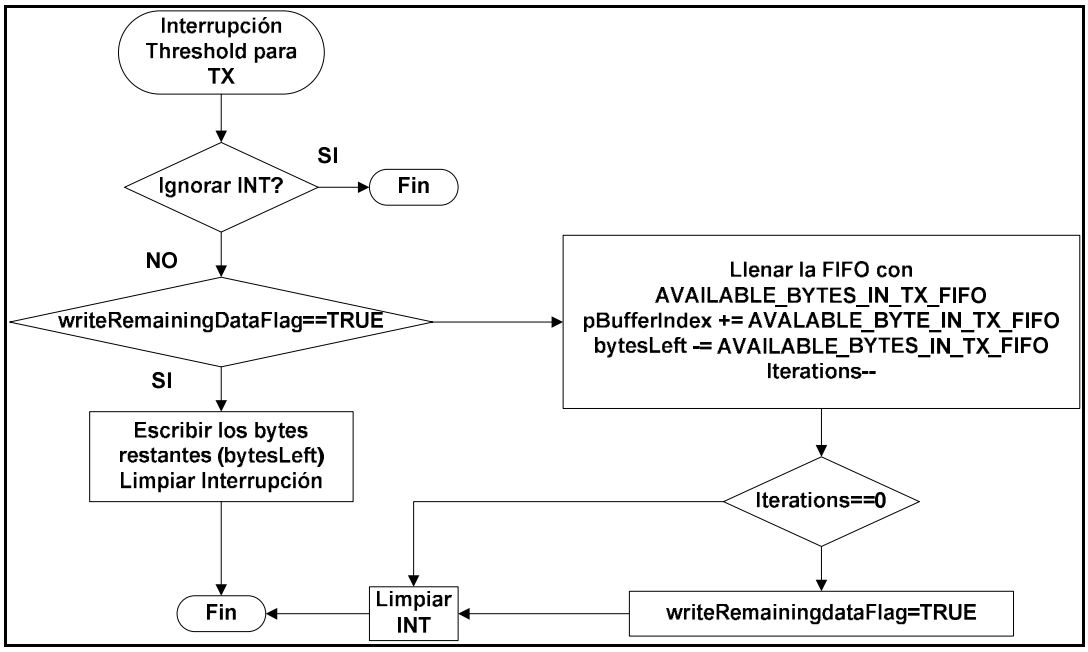


Figura 5.40 Interrupción de umbral (GDO2) en una transmisión.

En la interrupción de threshold se realiza cada recarga al FIFO_TX, en caso de ser necesario. La interrupción puede ser ignorada según se explicó anteriormente. También se controla si se deben de cargar los 33 bytes disponibles (AVILABLE_BYTES_IN_TX_FIFO), o si se escriben únicamente los bytes restantes. Por último se actualiza el valor de "iterations" y se utiliza para determinar la acción a tomar en la próxima interrupción.

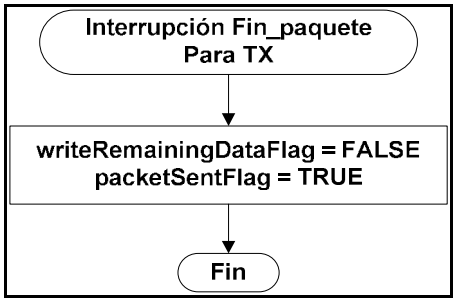


Figura 5.41 Interrupción fin de paquete (GDO0) en una transmisión.

La interrupción de fin de paquete es mucho más sencilla, solamente se actualizan las banderas de control y se notifica al control que la transmisión ha terminado, de forma que éste puede enviar otro paquete o regresar al transductor al estado de recepción.

5.2.4.3 Flujo de la información en el Ethmesh

El funcionamiento en general del Ethmesh es bastante fácil de entender ya que está diseñado como una sencilla máquina de estados con un control de flujo determinado por eventos, el uso de los sockets BSD facilitó el diseño ya que los sockets son revisados cada vez que la máquina de estados pasa por un estado específico, las solicitudes de transmisión y recepción se van almacenando para ser atendidas cuando les corresponda, las tareas relacionadas con los mensajes UDP se ordenan de manera automática en un sistema multitarea, que asigna la oportunidad de su ejecución por periodos determinados de tiempo. En la figura 5.42 se platea el funcionamiento general del software contenido en el Ethmesh mediante una máquina de estados que resume el lazo infinito de tareas.

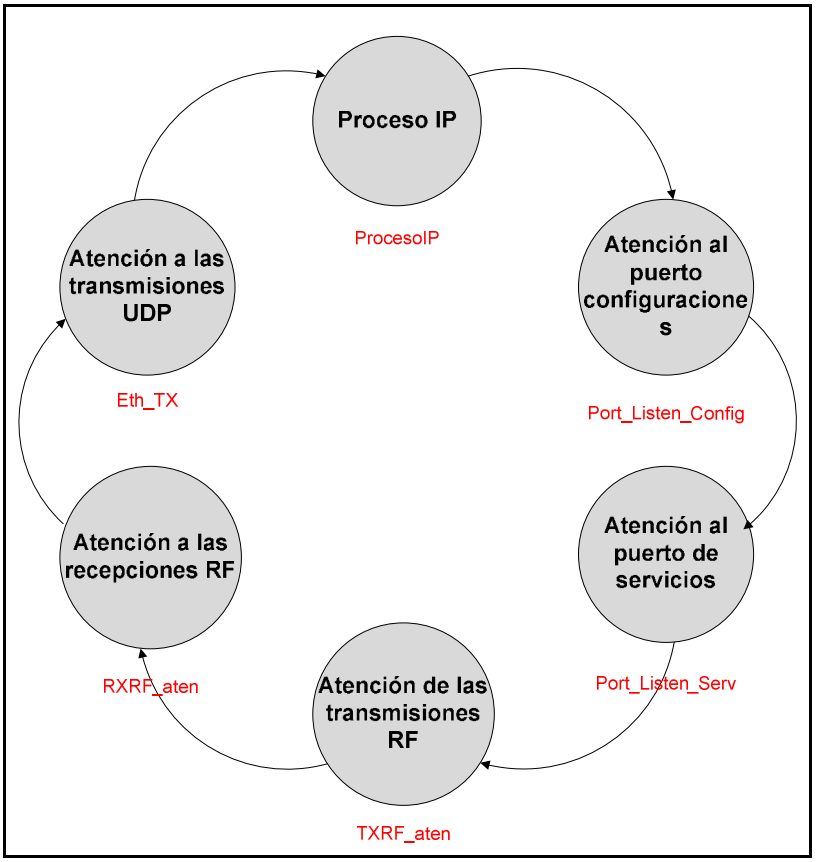


Figura 5.42 Diagrama de estados del control del Ethemesh.

La figura 5.42 sugiere que los seis estados del Ethmesh son recorridos, se omite la inicialización previa que, obviamente no se encuentra en el lazo de tareas, pero que sí es necesaria para levantar todo el sistema, en dicha inicialización se realizan las configuraciones básicas a la etapa de radiofrecuencia y al Stack de Microchip (véase la figura 5.31). La tabla 5.12 ofrece una lista de las funciones involucradas en cada estado.

Tabla 5.12 Funciones relacionadas a cada estado del Ethmesh.

Estado	Funciones	Descripción
ProcesoIP	TCPIPProcess()	Ejecuta la máquina de estados del stack: envíos pendientes del stack, ejecuta un tick, procesa recibidos.
Port_Listen_Config	Config_Receiver()	Administra las configuraciones con base en los mensajes que recibe por el puerto 6651.
	recvfrom(s)	Función BSD socket, acepta mensajes UDP desde cualquier IP y por el Puerto 6651.
Port_Listen_Serv	recvfrom(s3)	Acepta mensajes UDP por el puerto 6653, y retorna la dirección IP de la solicitud para que Port_admin_IN la registre en la tabla de enrutamiento.
	Port_admin_IN ()	Administra las entradas de datos desde la red Ethernet, lee el paquete proveniente de algún PCy según el contenido determina si debe responder una solicitud de conexión, una solicitud de máscara RF o si es una sencilla transmisión por radio. Además registra las direcciones IP entrantes que sean nuevas o desconocidas.(Actualiza la tabla de direcciones)
TXRF_aten	TX_RF_Start()	Comienza una Tx_RF. Como se supone que esta se ejecuta si no hay una recepción en progreso, se parte del hecho de que el cc1101 se encuentre en estado de IDLE. Retorna falso si se inició la transmisión y falso en caso de error.
	Clean_Array()	Limpia un buffer de determinado tamaño. Reside en el archivo CC1101control.c
	Config_GDO()	Configura la función del GDO0, recibe definiciones en CC1101control.h que están relacionadas al tipo de interrupción generada por el CC1101. Reside en CC1101control.c
	Strobe()	Mueve de estados al CC1101.
	Threshold_Polarity	Rutina de configuración del pin GDO0, selecciona una polaridad para la interrupción de threshold.

Tabla 5.12 Funciones relacionadas a cada estado del Ethmesh (continuación).

Estado	Funciones	Descripción
RXRF_aten	CRC_Check()	Rutina para la revisión del CRC de los paquetes recibidos por RF. Devuelve TRUE cada vez que encuentra un paquete cumple con el CRC.
	Config_GDO()	Configura la función del GDO0, recibe definiciones en CC1101control.h que están relacionadas al tipo de interrupción generada por el CC1101. Reside en CC1101control.c
	Strobe()	Mueve de estados al CC1101.
Eth_TX	Port_Admin_OUT()	Administrador de la salida de paquetes que provienen de RF y se direccionan hacia una IP por Ethernet.

Algunas de las funciones en la tabla 5.12 requieren de especial explicación porque en ellas se encuentra la lógica del funcionamiento del Ethmesh. Siguiendo el orden en que aparecen en la tabla, la primera función importante es *Config_Receiver()*. Esta función obedece a la lógica del diagrama de la figura 5.43.

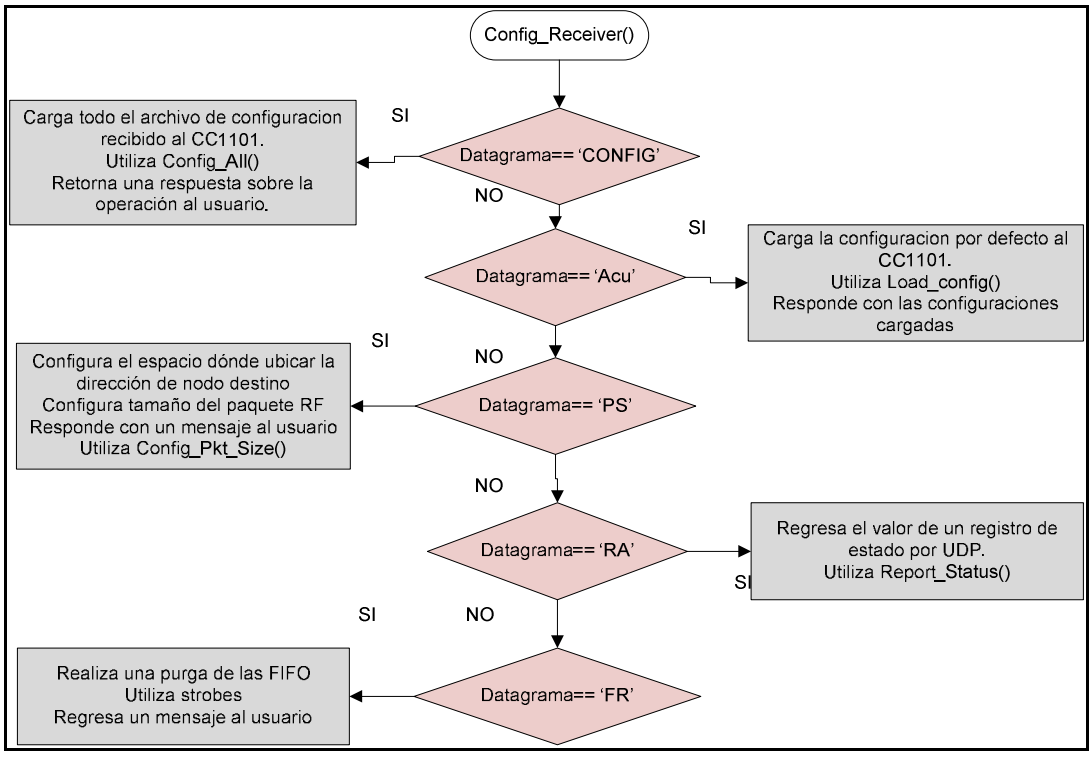


Figura 5.42 Diagrama de flujo de la función de recepción por puerto de configuración.

La siguiente función que merece atención es Port_admin_IN (). Esta función está vinculada con el registro de las direcciones IP en una tabla de enrutamiento, lee e interpreta los mensajes entrantes por el puerto 6653, y con base en el string leído invoca a la función de chequeo de tabla de enrutamiento o de carga al buffer interno del Ethmesh.

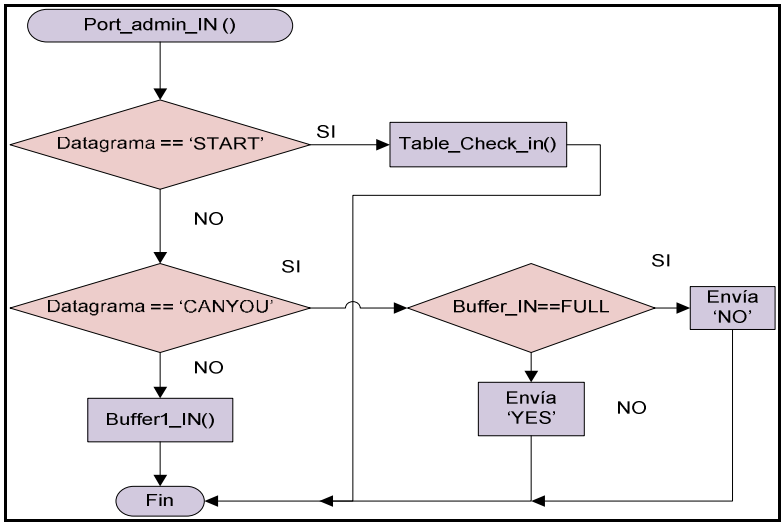


Figura 5.43 Atención de solicitudes y recepción de paquetes por el puerto 6653.

Como se ha mencionado en la primera sección de este capítulo, las computadoras en la red Ethernet envían un mensaje especial para solicitar al nodo Ethmesh que las anote en su tabla de enrutamiento, la figura 5.44 es una representación lógica que tiene la tabla de enrutamiento, se le llama representación lógica debido a que la tabla de enrutamiento es en realidad un arreglo de estructuras que almacenan tres datos para mantener una conexión, la dirección lógica de la computadora es usada para reenviar de vuelta los mensajes necesarios hacia la computadora indicada, de manera que se establece una conexión punto a punto entre un nodo en especial y una PC en la estación de control. Lo que se le llama máscara de RF es la dirección de nodo que el Ethmesh usará para representar a esa computadora que solicita la inscripción en la tabla, quiere decir que cualquier mensaje que tenga como destino el nodo 25, es en realidad un vehículo respondiendo un mensaje a la computadora con la dirección 192.168.2.101, y así sucesivamente se asocian todas las direcciones en una especie de mapeo. Por último el TTL es una abreviatura para, "time to live" que no tiene nada que ver con los saltos por un enrutador como pasa con los mensajes ICMP (ver apéndice A3). El TTL es más bien un tiempo que se preestablece para el enlace

entre una PC y un nodo. Es decir el tiempo de validez que tiene la dirección escrita en la tabla antes de ser reescrito por el control del Ethmesh. Una vez reescrito el mensaje no habrá forma de que los mensajes destinados al nodo 25, continuando con el ejemplo, lleguen hasta la primera máquina de la lista en la figura 5.44. Esta técnica de mantenimiento de direcciones se pensó para evitar una saturación en la tabla con direcciones inútiles que podrían siquiera estarse utilizando y que por ende, únicamente representarían un gasto innecesario de memoria en el microcontrolador del Ethmesh (en un buen diseño, se aprovechan al máximo los recursos disponibles del hardware con el que se cuenta).

Dirección IP	Máscara RF	TTL (min)
192.168.2.101	25	12
192.168.2.104	123	25
192.168.2.105	45	5
192.168.2.106	78	1

Figura 5.44 Tabla de enrutamiento para el Ethmesh

Ahora bien, retomando el diagrama en la figura 5.43se observa que la petición de inscripción a la tabla se lleva a cabo por medio de un mensaje “START”, cuando el control interno encuentra este mensaje procede a llamar a la función *Table_Check_in*. La tabla tiene un valor de 100, quiere decir que es capaz de manejar hasta cien enlaces simultáneamente, eso es mucho más de lo necesario ya que en la oficina de control se tienen unas 10 computadoras.

Como se puede ver en la figura 5.45, el algoritmo primero busca una dirección IP que coincida con la solicitante, esto se hace así para evitar inscribir una misma PC varias veces, en vez de gastar más espacios en la tabla se reescribe el TTL para esa PC y su tiempo de vida es refrescado, de esta forma una computadora puede mantener un enlace habilitado durante un tiempo indefinido si se mantiene enviando estas solicitudes cada cierto tiempo dependiendo del TTL inicialmente escogido.

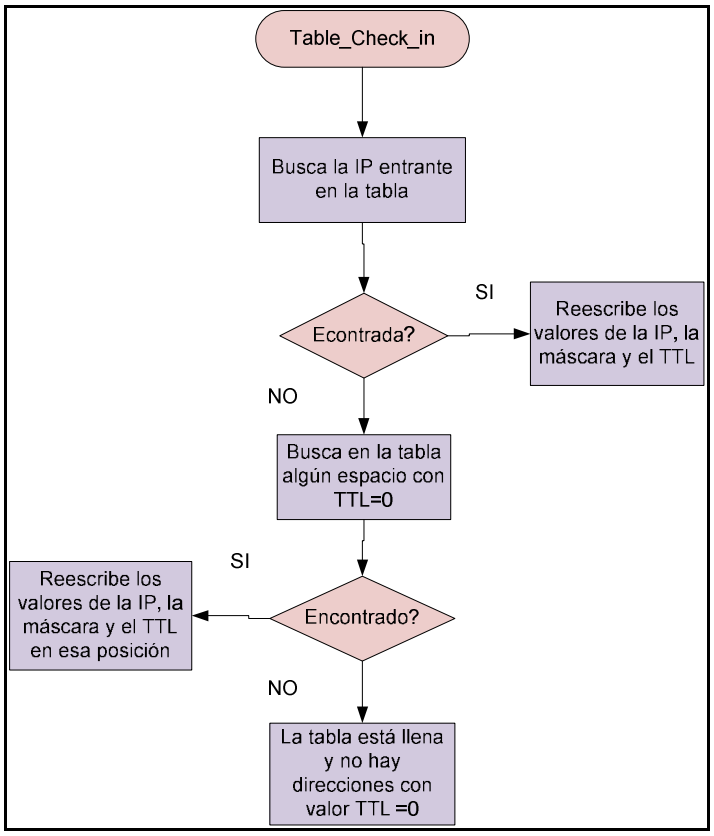


Figura 5.45 Algoritmo de inscripción en la tabla de enrutamiento.

Otro aspecto fundamental en la figura 5.43 es el Buffer_IN, este nombre se le asignó a la FIFO programada en el interior del Ethmesh, ésta tiene una capacidad de 1024 bytes y se utiliza para amortiguar el cuello de botella formado por la diferencia de velocidades entre las dos interfaces. La idea es que si varias computadoras comienzan a enviar datos hacia la red de RF, el Ethmesh tenga la capacidad de almacenar los paquetes mientras los envía por medio del CC1101. Para ello se dispone de una función que almacena los paquetes entrantes en esta FIFO y reporta cuando se llena dicha memoria. Este aviso a través de la variable booleana "BUFFER_FULL" es usado por la función de administración del puerto como criterio para responder a las solicitudes de envío de paquetes aceptación de paquetes. La solicitud de aceptación de paquetes es una especie de protocolo para regular el flujo de datos a través de la interfaz Ethernet que llegan al Ethmesh. La idea es que cada vez que un paquete quiera ser enviado por una PC, primero ésta envía un mensaje preguntando si el Ethmesh está en capacidad de aceptarlo para su posterior envío por radio. En caso de recibir una respuesta positiva, expresada como un

mensaje con la palabra “YES”, la PC solicitante envía el paquete como un mensaje UDP. En caso de no recibir una aceptación la computadora espera un tiempo aleatorio y vuelve a preguntar, este es un algoritmo similar al utilizado por las computadoras conectadas a una red Ethernet para solucionar colisiones (ver 3.5.2). *Buffer1_IN()* también actualiza la posición del índice de escritura en la FIFO del Ethmesh, de esta forma no se reescribe sobre otros datos y se mantiene un orden en los accesos a esta memoria. La memoria BUFFER_IN es controlada por medio de dos índices, uno apunta al final y es usado para las escrituras (index_End) y otro que es usado para las lecturas (index_Start), en el momento en que ambos se igualan se inicializan a la posición origen, de esta manera se obtiene una especie de arreglo circular donde se van almacenando los paquetes a transmitir.

La siguiente función relacionada con la administración de los puertos es *Port_Admin_OUT()*, dicha función realiza un pequeño procesamiento sobre los paquetes recibidos por radio y almacenados temporalmente en el buffer “rxBuffer”. Dicho procesamiento consiste en la extracción del tamaño del paquete recibido por RF y pendiente a enviar como mensaje UDP. Además se establece en la variable Dir_rf la dirección de nodo destino, con base en la posición indicada por la variable configurable “Mask_Location”, el administrador de puerto de salida decide si el paquete recibido es un mensaje de multidifusión y debe ser retransmitido a todas las PC´s en la red de Ethernet. En caso de no ser un broadcast, se procede a buscar la dirección del nodo a en la tabla de enrutamiento y enviar el mensaje a la IP correspondiente a través del puerto 6652.

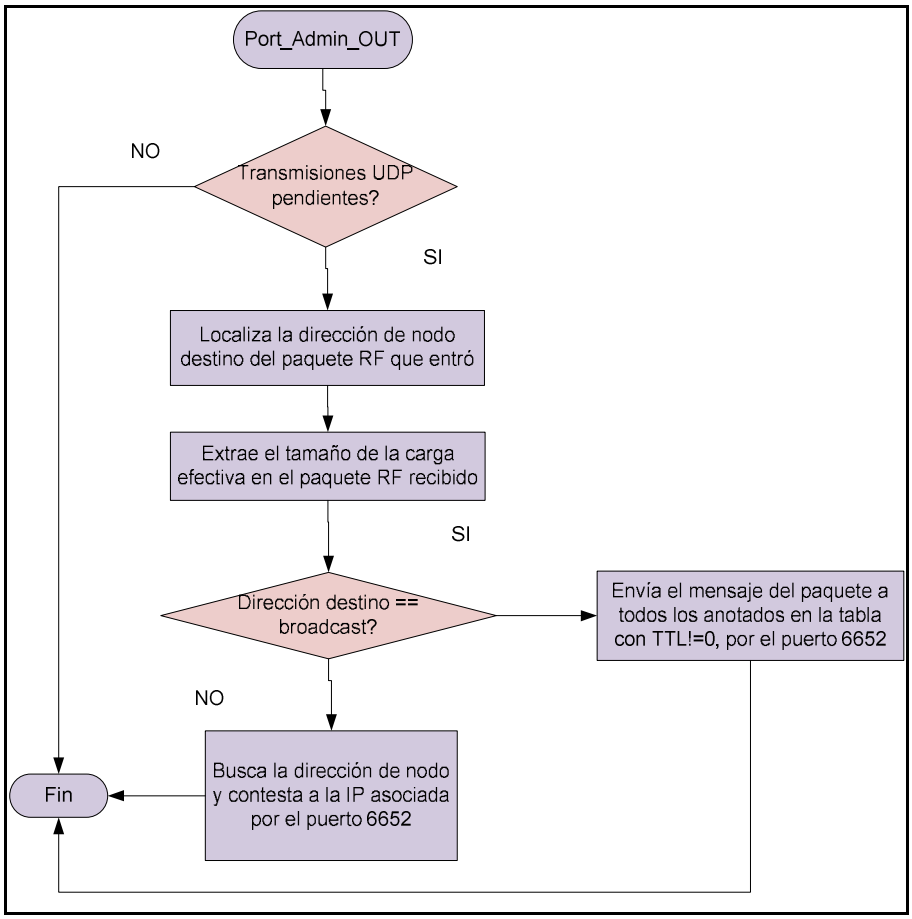


Figura 5.46 Administrador de envío de datagramas.

Retomando la tabla 5.12, la siguiente función por analizar es TX_RF_Start(), sin embargo ya se ha hablado de esta función en la sección 5.2.4.2 y en la figura 5.39 aparece el diagrama de flujo de su funcionamiento. Pero es importante mencionar que las banderas o variables booleanas usadas para el control del flujo fueron clave para el adecuado funcionamiento de la máquina de estados del Ethmesh, en las figuras de la sección 5.2.4.2 aparecen en color rojo y con mayúsculas. En la siguiente tabla se hace un recuento de dichas banderas de control y su función en el control del Ethmesh.

Tabla 5.13 Banderas para el control de flujo en el software del Ethmesh.

Variable	Estado inicial	Descripción
PRIMERO	FALSE	Usada para la detección en el error de la interrupción de fin de paquete en las recepciones RF.
RX_PROGRESS	FALSE	Usada para monitorear cuándo una recepción está en proceso, de ser true no es posible comenzar con una transmisión
ERROR_RX	FALSE	Usada para la corrección en el error de la interrupción de fin de paquete en las recepciones RF. Cuando se hace true se procede a reconfigurar las GDOx del CC1101.
TX_FINISHED	FALSE	Usada para controlar cuándo se ha terminado en caso de ser verdadera se debe reconfigurar la polaridad para GDO0 y retornar a RX.
TX_PROGRESS	FALSE	Se usa para confirmar la disponibilidad del CC1101, es verdadera mientras no se haya completado una transmisión.
ERROR_TX	FALSE	Usada para detección de errores en la transmisión, se hace verdadera cuando el CC1101 no entra en estado TX cuando se desea. Provoca una reconfiguración de los GDOx y un purgado de las FIFO.
Ignorar_Threshold	FALSE	Usada en transmisiones de paquetes de tamaño menor a 64bytes para ignorar la interrupción de recarga.
writeRemainingDataFlag	FALSE	Notifica el momento en que la recarga es hecha con los bytes restantes.
BUFFER_FULL	FALSE	Indica si el Buffer del Ethmesh se encuentra lleno y no es posible aceptar paquetes.
BUFFER_EMPTY	FALSE	Usada para controlar el momento de generar una transmisión RF.
ETHTX	FALSE	Se hace cierta si un paquete pasa el CRC, notifica si se debe transmitir un mensaje UDP.
CRC_OK	FALSE	Se hace verdadero si el CRC por software es igual al calculado por el emisor.
packetReceivedFlag	FALSE	Se hace verdadera al terminar de recibir un paquete en la interrupción del GDO2.

Resumiendo todo lo anterior en pocas palabras, el software programado en el Ethmesh le permite realizar la distribución de paquetes UDP hacia la radio mediante una máquina de 6 estados donde 3 de ellos son para la atención de los puertos y ejecución de tareas de conectividad de red, otros 2 funcionan para lidiar con transmisiones y recepciones de la etapa de RF construida y por último un estado que verifica la necesidad de responder a las computadoras en la LAN con mensajes UDP. Como ya se mencionó el puerto 6651 permite varias opciones de configuración que se utilizan para variar los parámetros del CC1101 y por ende, permite trabajar en otras redes que siempre y cuando se encuentren en el rango de 315-433-MHz, eso por el dimensionamiento físico de los componentes. Por otro lado el puerto 6652 es usado para recibir solicitudes de inscripción en la tabla de enrutamiento con mensajes que contienen una máscara de RF deseada, dicha máscara será usada para asociar las respuestas de la radio a una PC en especial. También puede que por este puerto se reciban mensajes, si hay espacio en el buffer interno, para ser retransmitidos hacia algún nodo en especial. Por último el puerto 6552 es el puerto por el cual se envían los paquetes capturados desde los camiones a una PC según la dirección que contenga una posición ajustable en el paquete o bien, puede ser una solicitud de broadcast desde algún nodo RF que debe ser retransmitida a todos los computadores cuyas direcciones IP se encuentran anotadas en la tabla de enrutamiento y que cuentan con un TTL distinto de cero, o sea que su tiempo de enlace no ha expirado.

Un detalle que falta mencionar es que para el refrescamiento de la tabla se utilizaron los temporizadores 4 y 5 en un modo especial que junta ambos registros de conteo, de forma que se permiten suficientes cuentas para que el reloj de 80MHz pre-escalado a 1/256 pueda generar cuentas de forma que, se tenga una interrupción cada minuto. Entonces en la interrupción generada cada minuto por los temporizadores 4 y 5 se realiza la resta de una unidad a todos los valores TTL diferentes de cero contenidos en la tabla de enrutamiento.

Por último es necesario aclarar que el software de prueba para la PC fue desarrollado con las herramientas descritas en la primera sección de este capítulo, y que no se pretende entrar en detalle de su concepción ya que no se considera parte de la solución, sino más bien una herramienta que forma parte de la metodología de prueba. Además sus funciones se ilustran en las pruebas realizadas que se documentan en el siguiente capítulo de resultados experimentales. El software de prueba que reside en la PC si bien es básico para el funcionamiento del Ethmesh, no se encuentra dentro de los objetivos de este proyecto y es completamente sustituible por alguna otra programación que sea capaz de manejar sockets para UDP, siempre y cuando cumpla con los requisitos de estar direccionado a los puertos adecuados y sea programado por alguien con un entendimiento sobre el funcionamiento del Ethmesh. Para dejar abierta la posibilidad de diseñar otros programas, el Ethmesh se diseño de la manera más flexible y sin dependencias del software en el PC, además se entregó un manual de usuario que resume los detalles pertinentes para el eventual diseño de un nodo virtual que ofrezca facilidades de interfaz grafica y otros detalles extras que se quieran incorporar.

Capítulo 6: Análisis de Resultados

6.1 Resultados Experimentales

En este capítulo se presenta en dos secciones, en la primera sección se muestran algunas mediciones para la etapa de radiofrecuencia y en la segunda sección se encuentra la documentación relacionada a todas las pruebas realizadas para probar el funcionamiento del Ethmesh.

6.1.1 Resultados para la etapa de Radiofrecuencia

Las siguientes mediciones se llevaron a cabo con el equipo analizador de espectros Agilent E4402B, este modelo es capaz de medir espectros que se encuentran desde los 9kHz hasta los 3GHz. Las mediciones se realizaron de dos formas, la primera fue usando antenas monopolo de λ/8 en el nodo Ethmesh y a la entrada del Agilent. En la segunda prueba se conecta el Ethmesh directamente a la entrada del equipo para medir la potencia de transmisión que emite el Ethmesh.

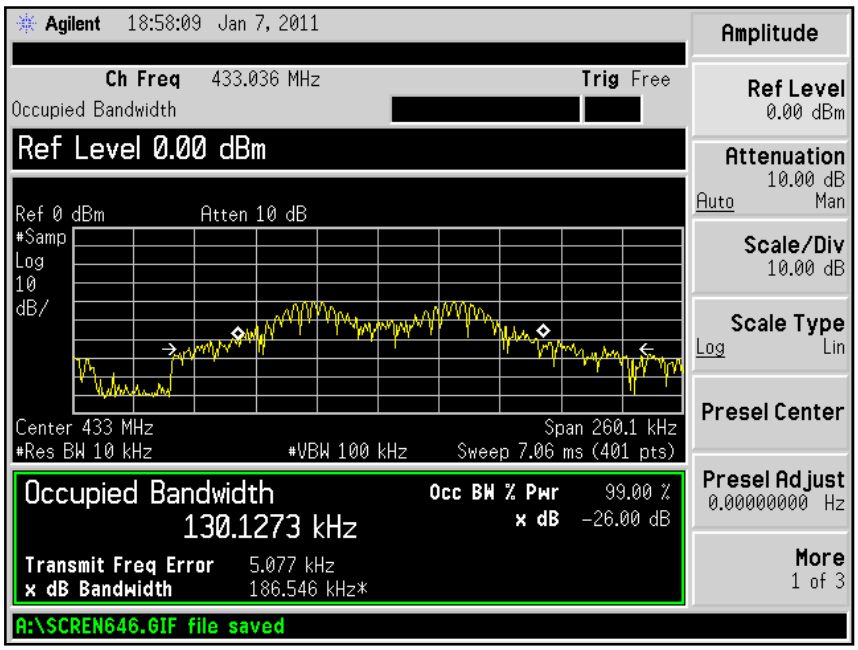


Figura 6.1 Medición del ancho de banda, usando antenas monopolo.

En la figura 6.1 se muestra el resultado de una medición de ancho de banda, para llevar a cabo esta medición se hizo una transmisión continua de un byte con valor 0xAA. Posteriormente se mide la frecuencia central y se compara con la frecuencia esperada para los valores cargados en los registros (usando como referencia el RFStudio).

Tabla 6.1 Mediciones de frecuencia central utilizando antenas de λ/8.

Frecuencia central medida [MHz]	Frecuencia central esperada [MHz]	Porcentaje de error [%]
433.04	432.999817	0.009

Con base en la frecuencia central medida se toman las mediciones de las frecuencias de desviación que se presentan en la tabla 6.2.

Tabla 6.2 Mediciones de la desviación de frecuencia hacia ambas direcciones.

Frecuencia	Valor medido [MHz]	Valor esperado [MHz]	Porcentaje de error [%]
Frecuencia marca	433.0701	433.0717383	0.0003
Frecuencia espacio	433.008	433.0082617	0.00006

Posteriormente se realizaron las mediciones de potencia, primero la salida del Ethmesh conectada directamente con el equipo de medición y luego con antenas monopolo de λ/8 a una distancia de separación de aproximadamente 20 metros.

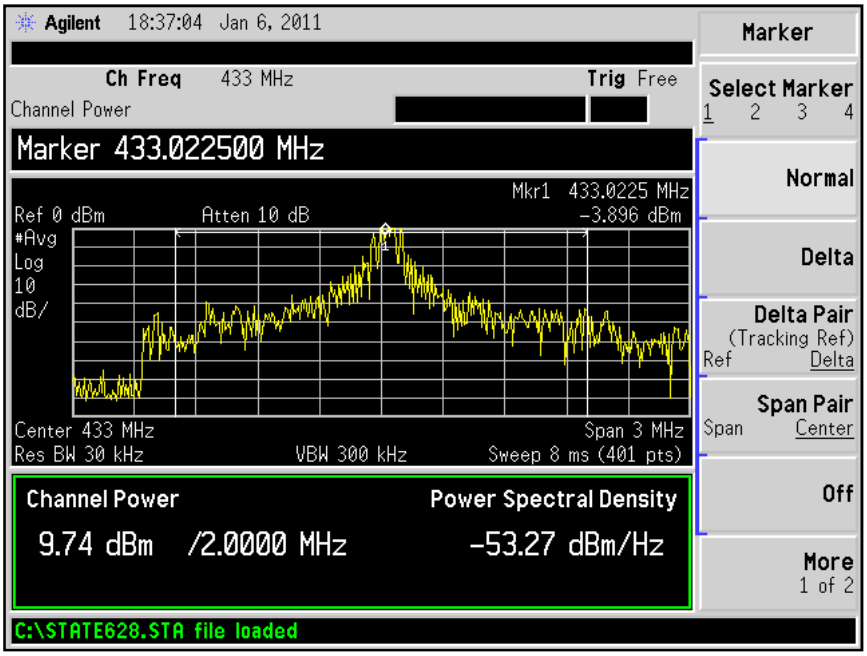


Figura 6.2 Medición de la potencia de salida del Ethmesh. Conexión directa.

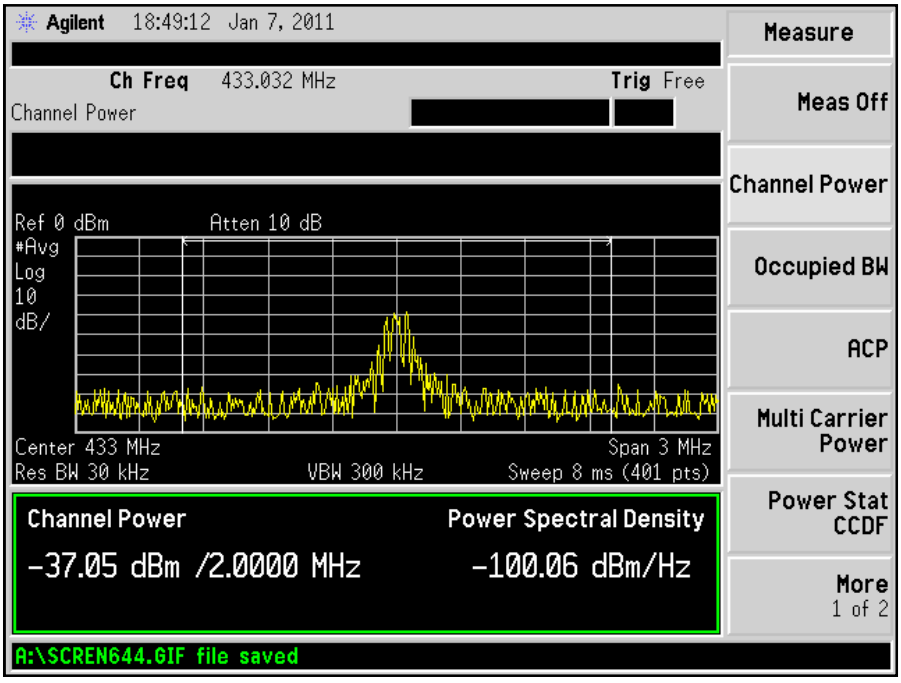


Figura 6.3 Medición de la potencia de salida del Ethmesh. Utilizando antenas.

6.1.2 Resultados en las pruebas de funcionamiento.

Luego hacer algunas mediciones del espectro se realizaron algunas mediciones con el analizador digital Agilent MSO7104A para corroborar la velocidad de transmisión de los datos en el canal de 433MHz. Estas mediciones se hicieron de forma digital utilizando los cursores del equipo Agilent, el procedimiento para la medición consistió en escribir paquetes de 11bytes al FIFO de de transmisión del CC1101, usando una interrupción de umbral que se activa cuando hay 9bytes en el buffer y que se desactiva cuando hay 8 bytes y la interrupción que se levanta cuando se termina de enviar la palabra de sincronización y el preámbulo y se desactiva al final del paquete. De esta manera fue posible establecer el tiempo que el CC1101 tarda en transmitir el preámbulo y la palabra de sincronía, además conociendo la cantidad de bytes para el paquete se puede medir también el tiempo dispensado en enviar un solo byte. Las mediciones se resumen en la tabla 6.3.

Tabla 6.3 Medición digital del tiempo relacionado a las transmisiones.

Intervalo de tiempo	Cantidad total de bytes [B]	Tiempo medido [ms]	Tiempo esperado según el bitrate [ms]	Porcentaje de error [%]
Envio del preámbulo y la palabra de sincronización	26	11	10.83	1.56
Envío del paquete	11	4.65	4.5837	1.42
Envío de un solo byte	1	0.4227	0.4167	1.44

En la siguiente sección se muestra el resultado obtenido al cargar los registros de configuración al Ethmesh utilizando la escritura en modo burst, las pruebas se realizaron con el equipo Agilent MSO7104A conectado a los puntos de prueba de la placa de radio.

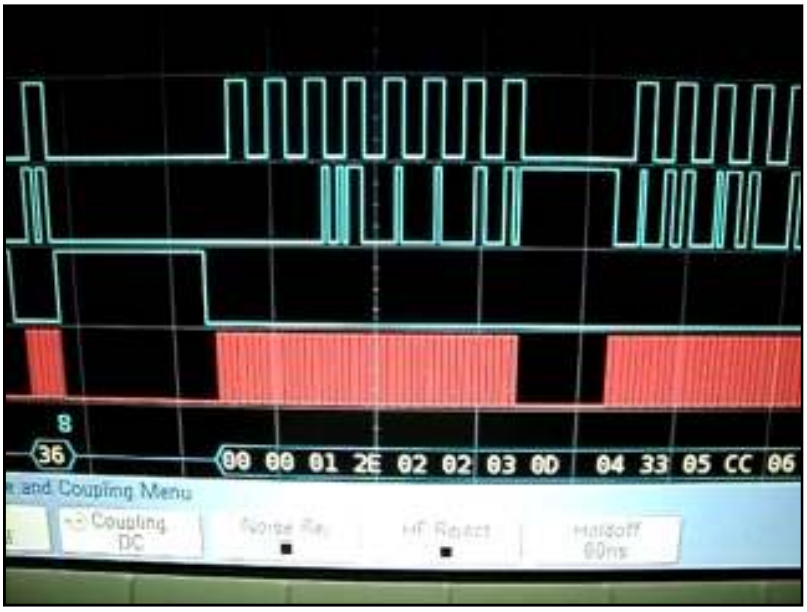


Figura 6.4 Escritura de los 5 primeros registros de configuración por acceso simple.

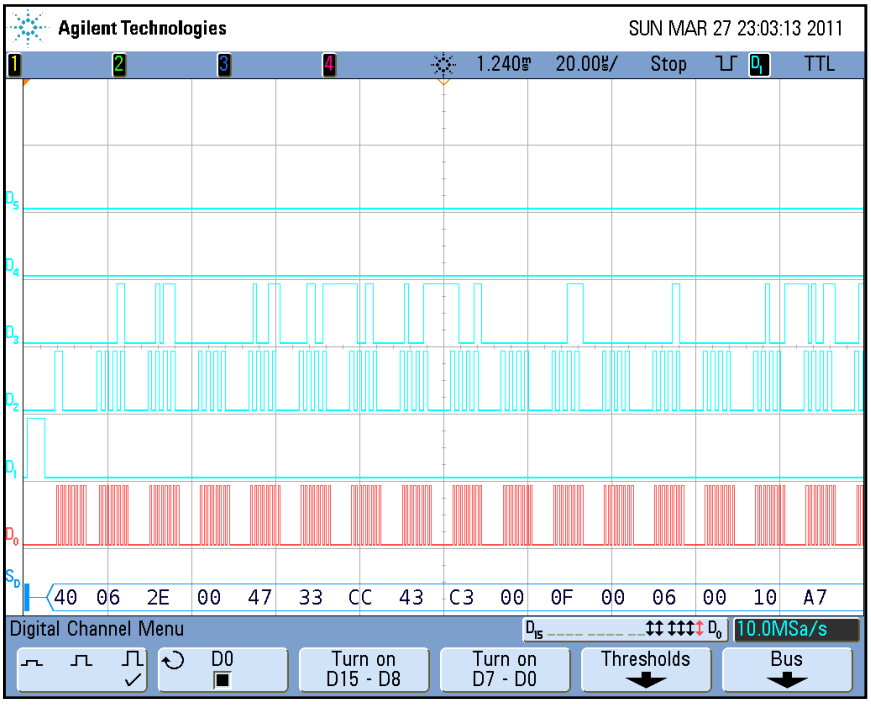


Figura 6.5 Escritura de los primeros 15 registros de configuración usando acceso burst.

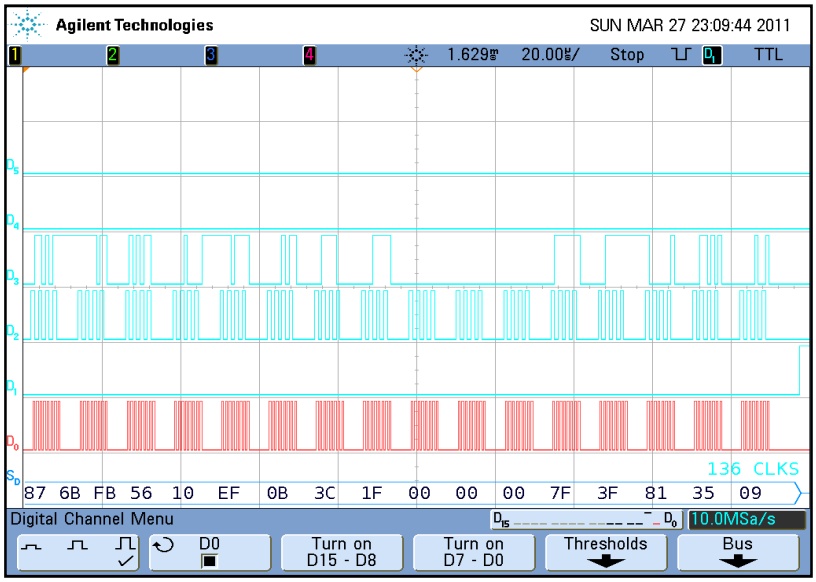


Figura 6.6 Escritura de los últimos 17 registros de configuración usando acceso burst.

En las figuras 6.5 y 6.6 se muestran los registros de configuración cargados hacia el CC1101 que fueron recibidos por medio de la interfaz de red desde un archivo de configuración.

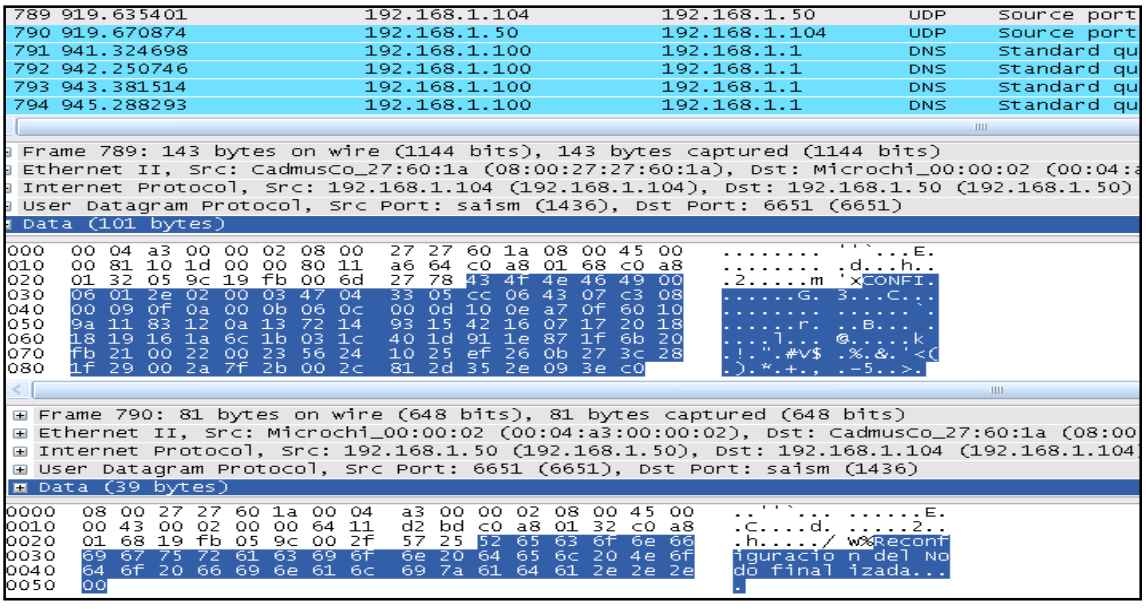


Figura 6.7 Envío del archivo de configuración, visualizado en Wireshark.

Posteriormente se realiza el envío de las configuraciones desde un archivo de texto por medio de la opción del menú de configuraciones del Ethmesh (ver figura 6.7). Utilizando la herramienta Wireshark se pueden capturar los paquetes en interés, la figura muestra el envío de los datos a cargar en los registros y la respuesta de confirma recibida desde el Ethmesh.

En el anexo B.7 se presenta el acceso al menú para cargar las configuraciones desde un plantilla que se presenta en B.8. Otra forma posible de hacer las configuraciones al Ethmesh es directamente desde el menú del software en la PC, en la figura 6.8 se visualiza el resultado luego de enviar una instrucción "AcuMine" al puerto 6651. Por otro lado en B.9 se presenta la opción en el menú principal de configuración así como el mensaje de respuesta después de enviar el comando.

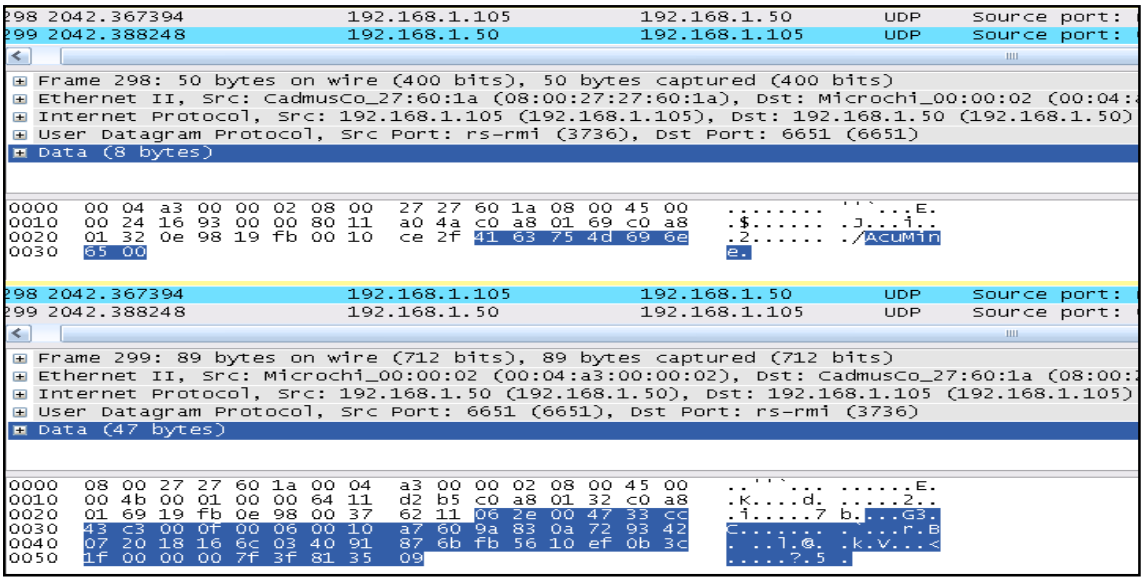


Figura 6.8 Envío de la instrucción de carga de configuraciones por defecto y retorno de datos por UDP.

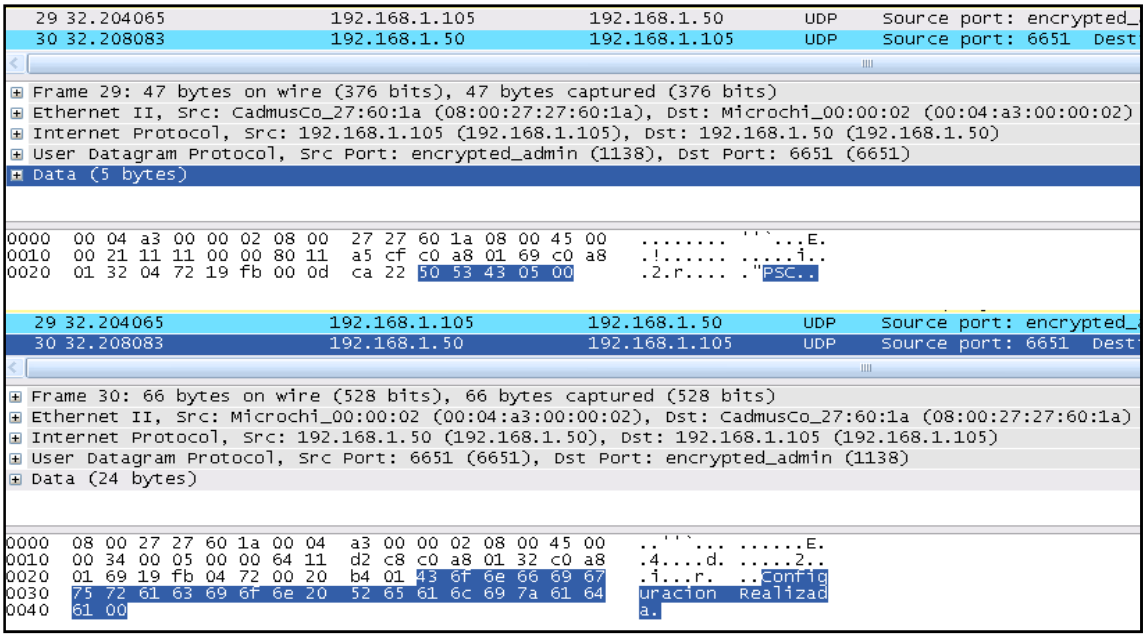


Figura 6.9 Envío de la instrucción de configuración de paquete y ubicación de la dirección para el nodo destino.

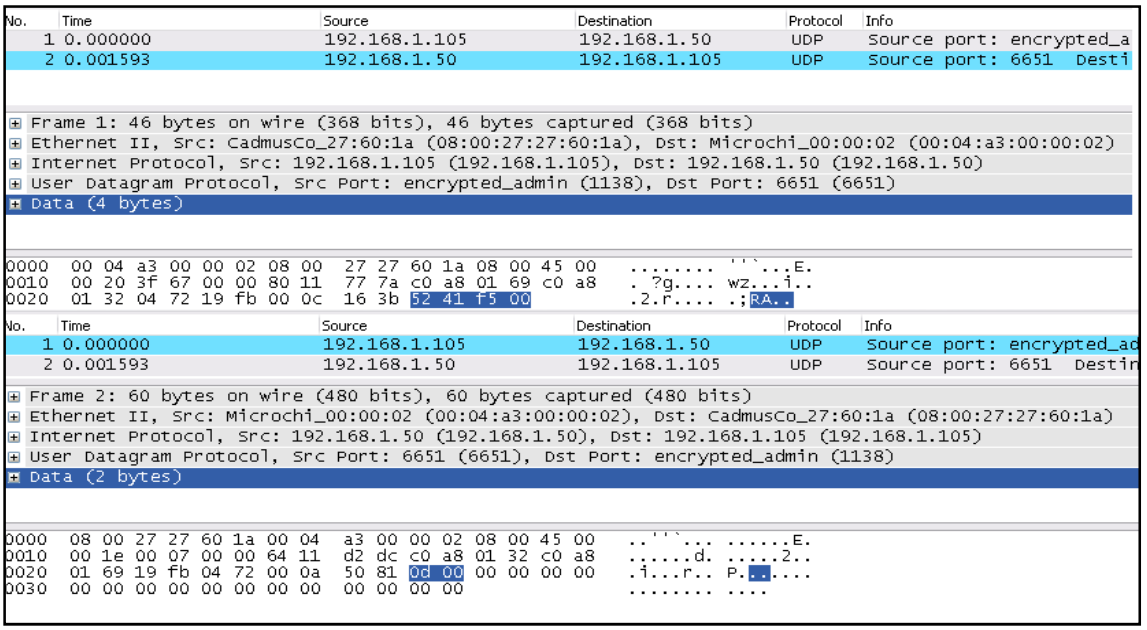


Figura 6.10 Lectura remota de un registro de estatus del CC1101 (estado de la máquina).

En las siguientes imágenes se muestran los resultados obtenidos al realizar una transmisión desde una PC al Ethmesh para que éste la almacene y posteriormente la envíe por radiofrecuencia.

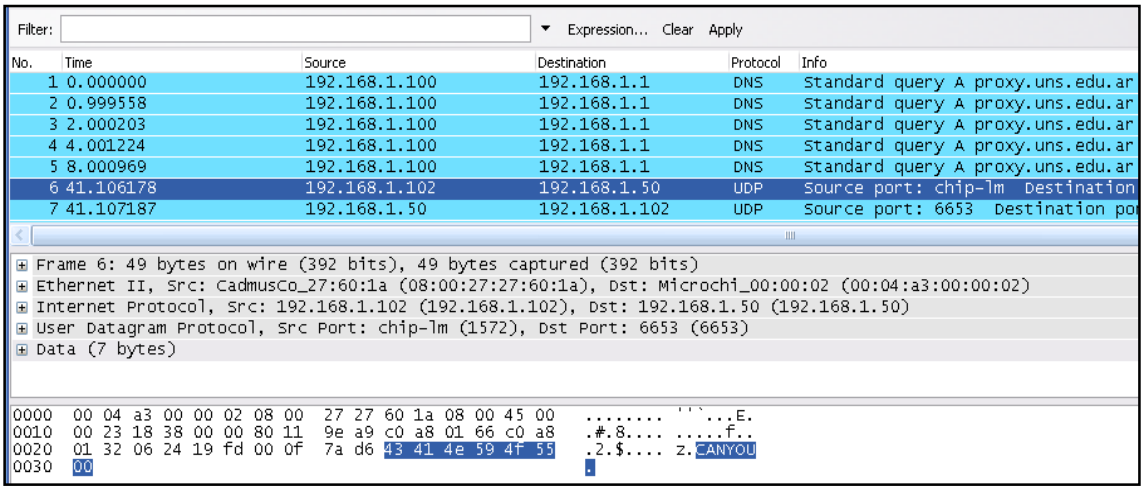


Figura 6.11 Envío de solicitud para transmitir por parte de una PC en la red Ethernet.

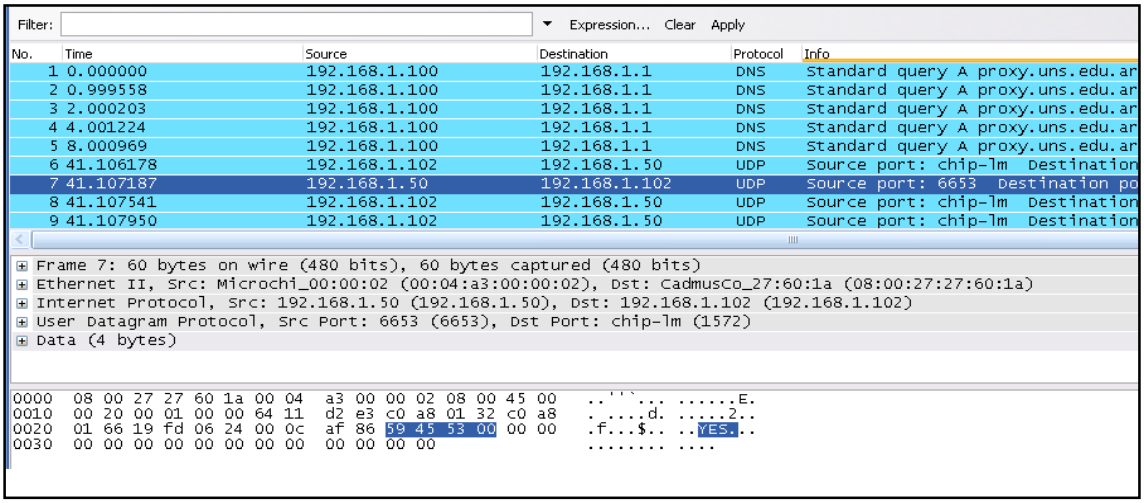


Figura 6.12 Respuesta desde el Ethmesh a la solicitud de transmisión.

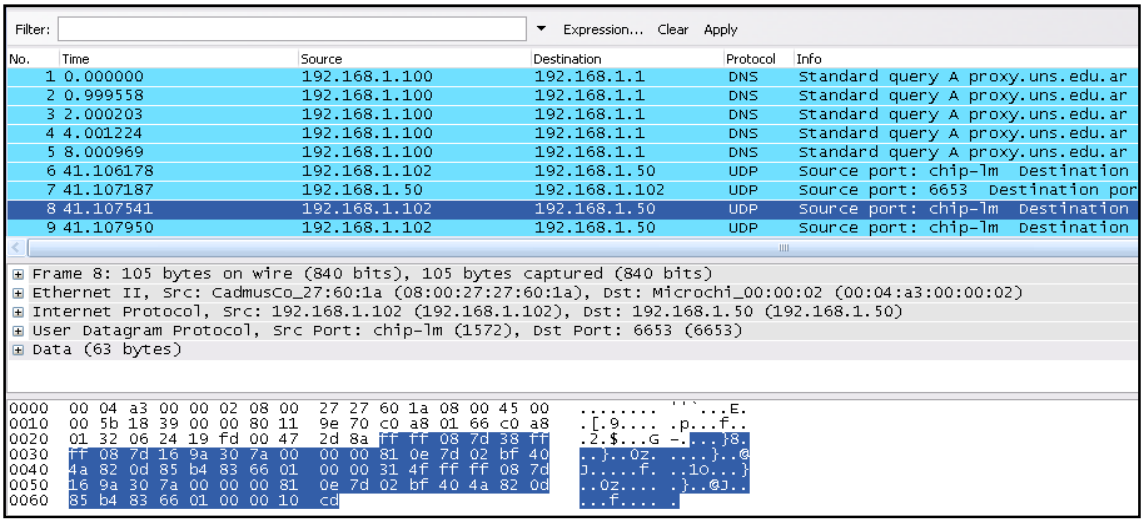


Figura 6.13 Envío de la PC hacia el Ethmesh una vez aceptada la solicitud.

Las figuras que se presentan a continuación corresponden a la inscripción de una IP a la tabla seguida de una recepción y después de un determinado tiempo la expiración del enlace.

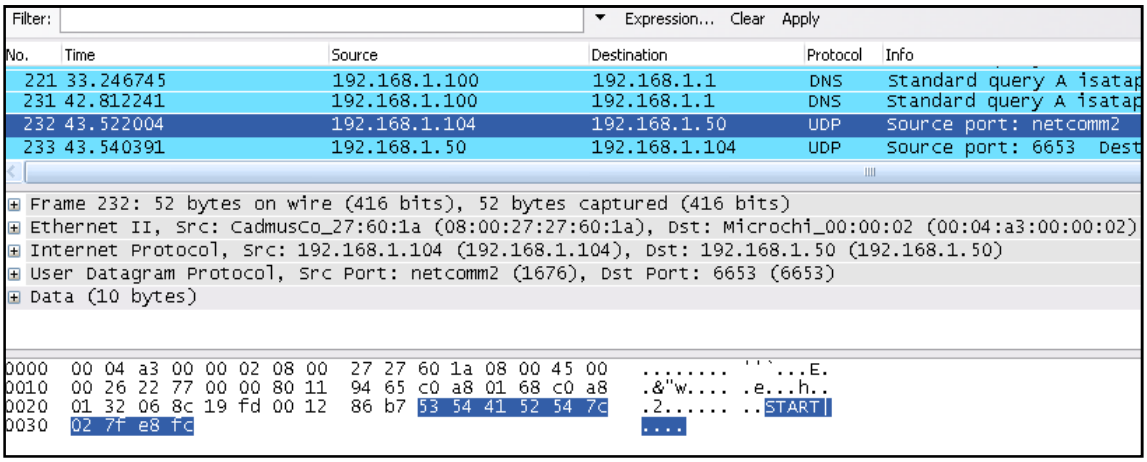


Figura 6.14 Envío de solicitud de inscripción desde la dirección 192.168.1.50.

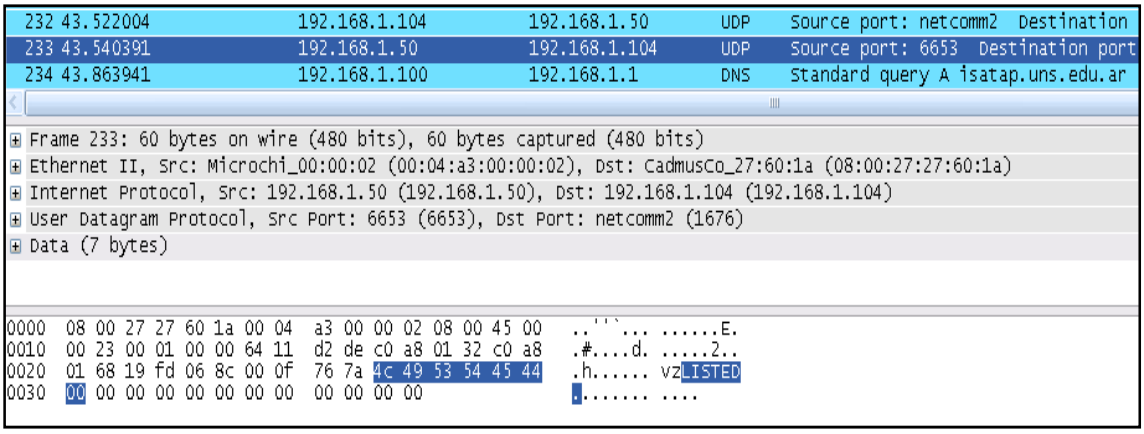


Figura 6.15 Respuesta de dirección anotada desde el Ethmesh.

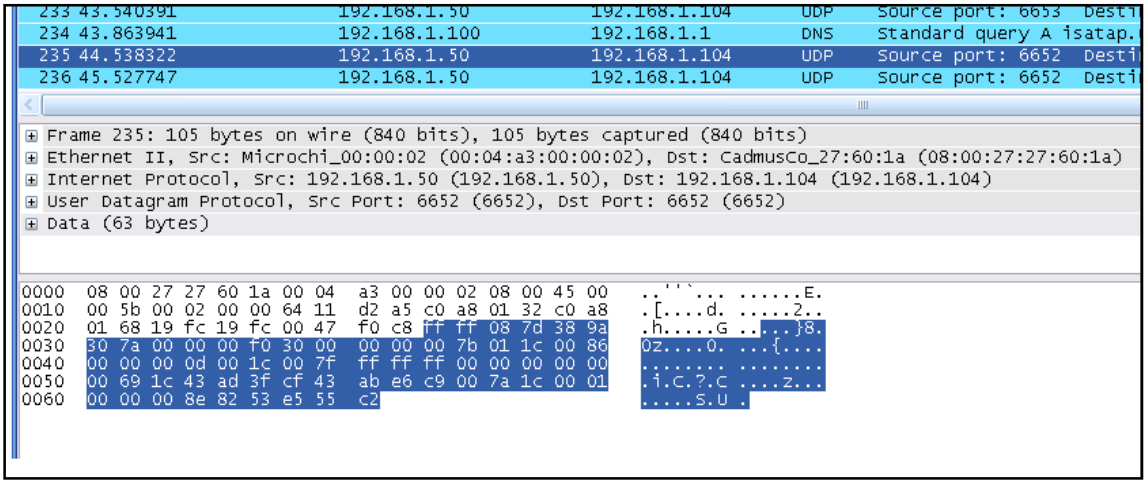


Figura 6.16 Primera recepción de paquete luego de inscrita la IP.

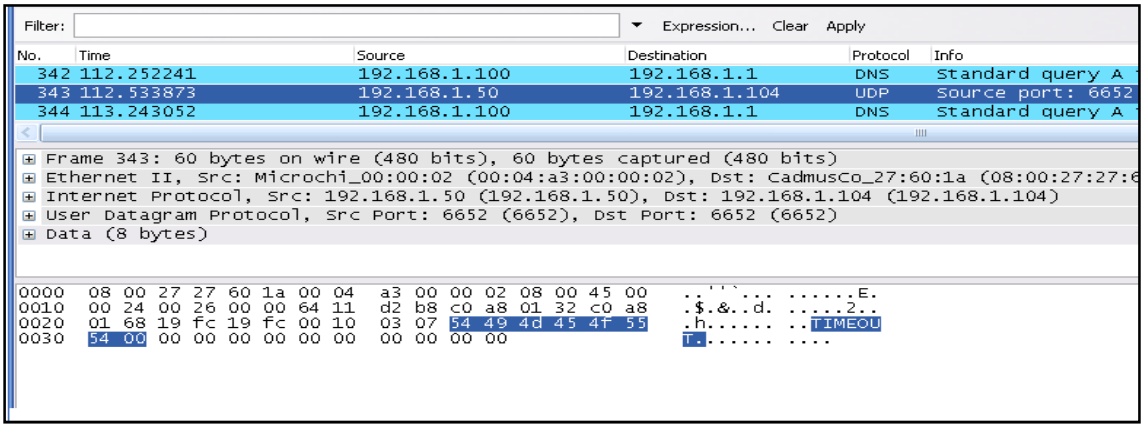


Figura 6.17 Notificación de tiempo de enlace expirado desde el Ethmesh.

Además se utiliza la herramienta de "watch" del MPLAB para verificar la inscripción de la IP en la tabla de enrutamiento.

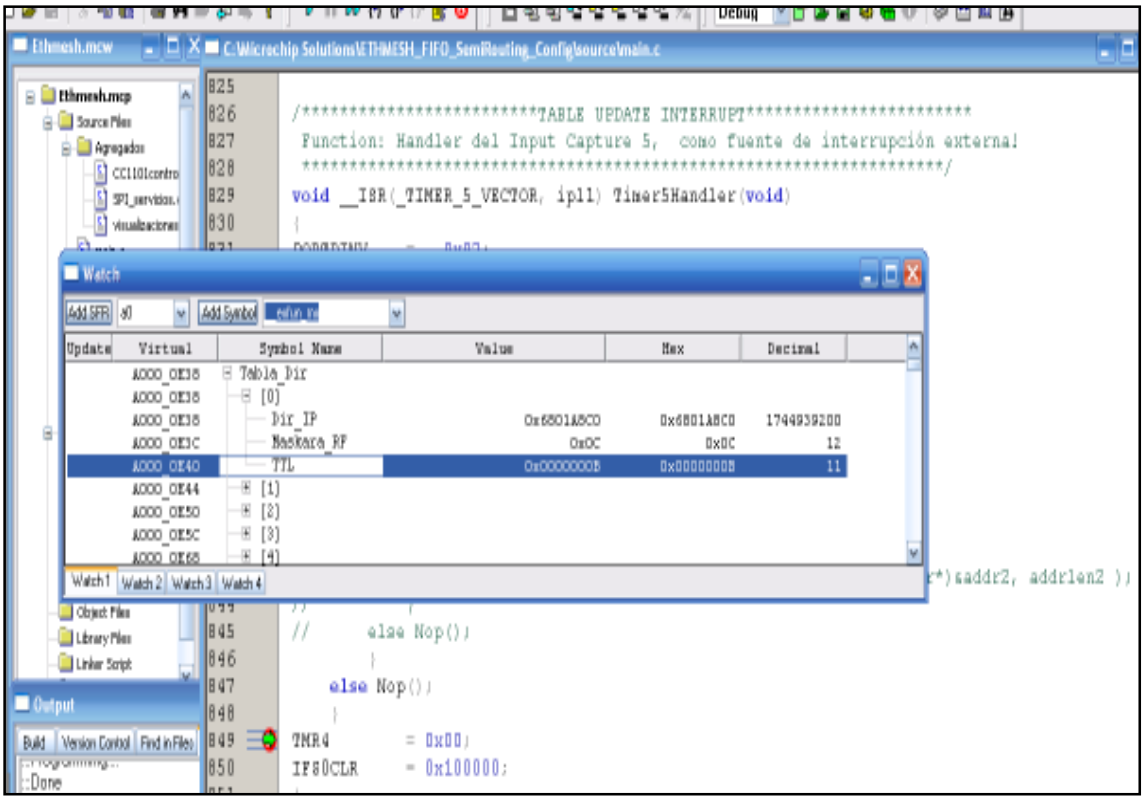


Figura 6.18 Verificación de la inscripción de una IP en la tabla de enrutamiento.

Para finalizar se hizo uso nuevamente del equipo Agilent MSO7104A para medir las señales que pasan por la interfaz SPI, entre el microcontrolador y el transductor de radio, esta fue una forma de verificar la correcta transmisión y recepción por para la etapa de radio.

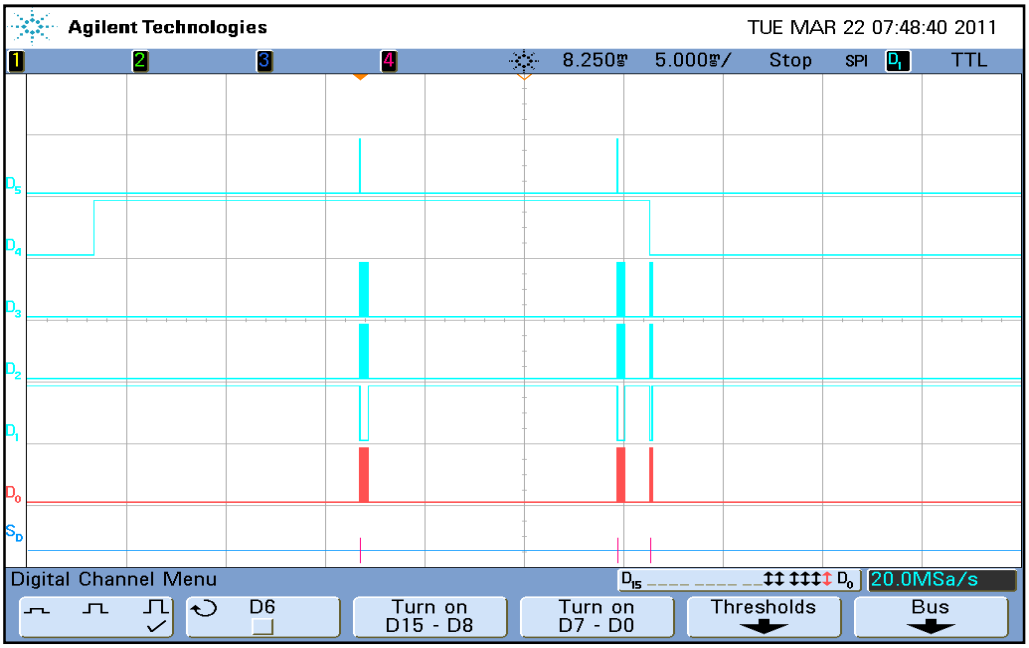


Figura 6.19 Recepción completa de un paquete de 67 bytes.

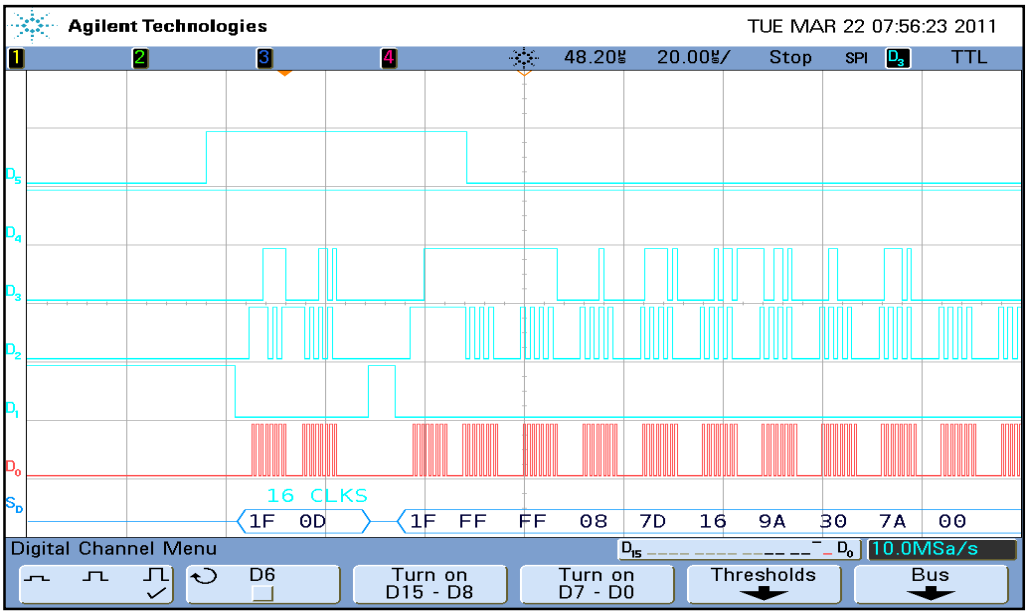


Figura 6.20 Primera lectura en interrupción de Threshold.

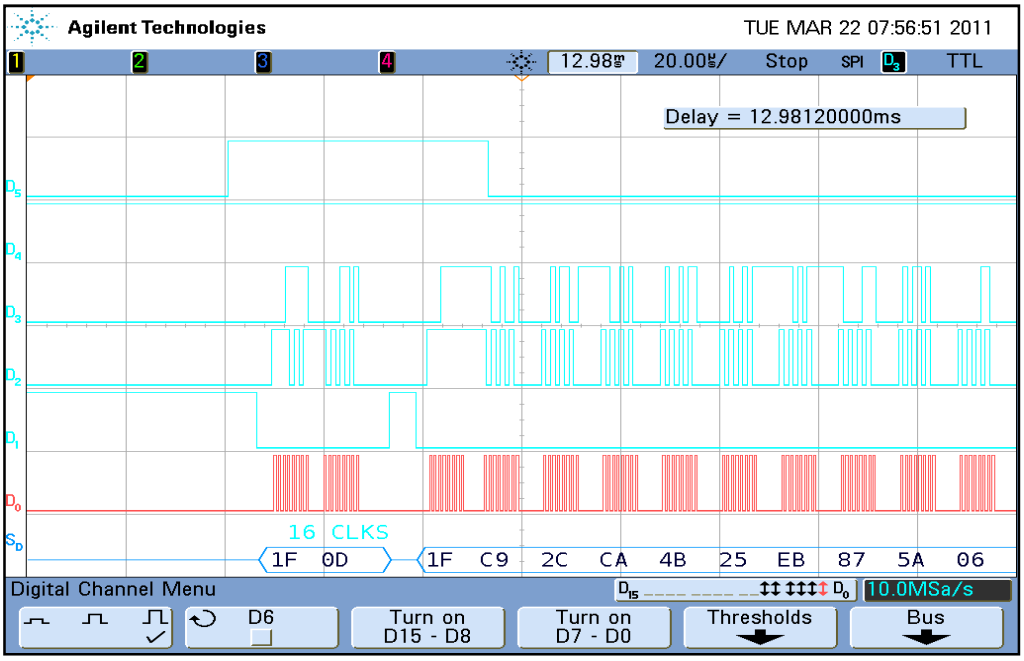


Figura 6.21 Segunda lectura en interrupción de Threshold.

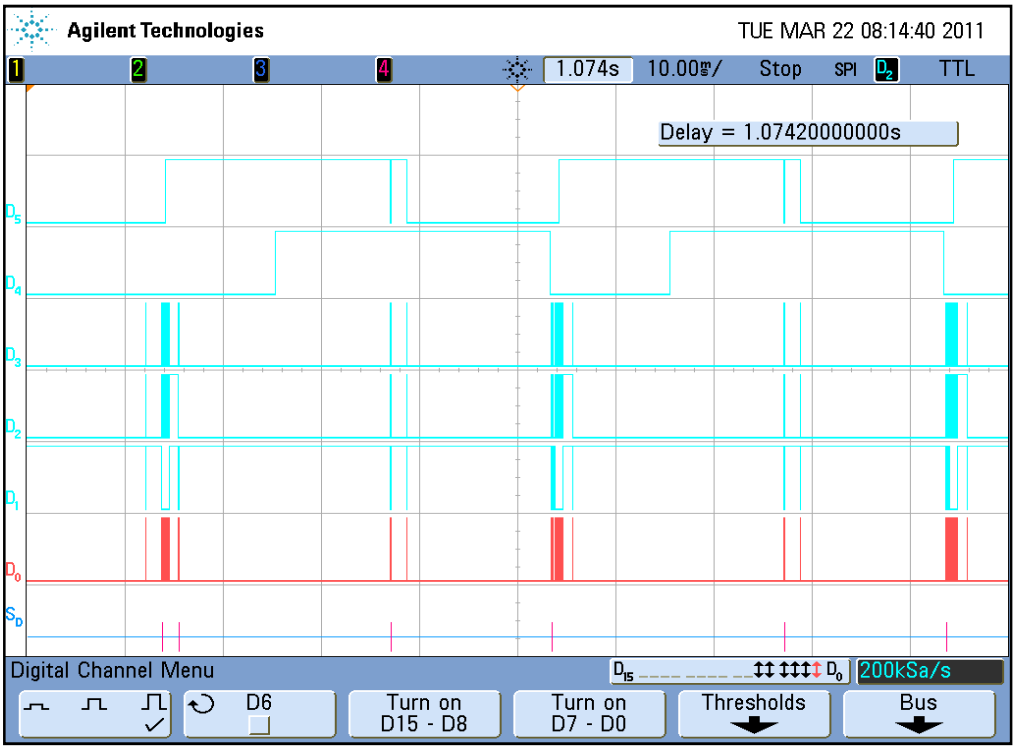


Figura 6.22 Operaciones de transmisión de paquetes de 67 bytes por radio.

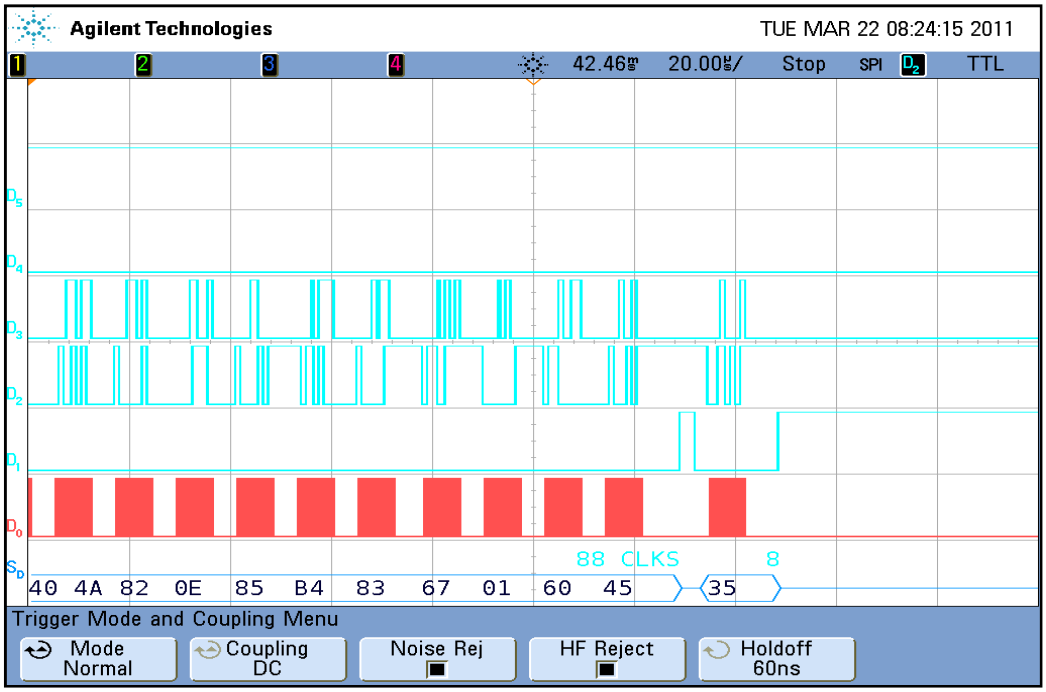


Figura 6.23 Escritura de últimos 10 bytes antes de envío a transmisión.

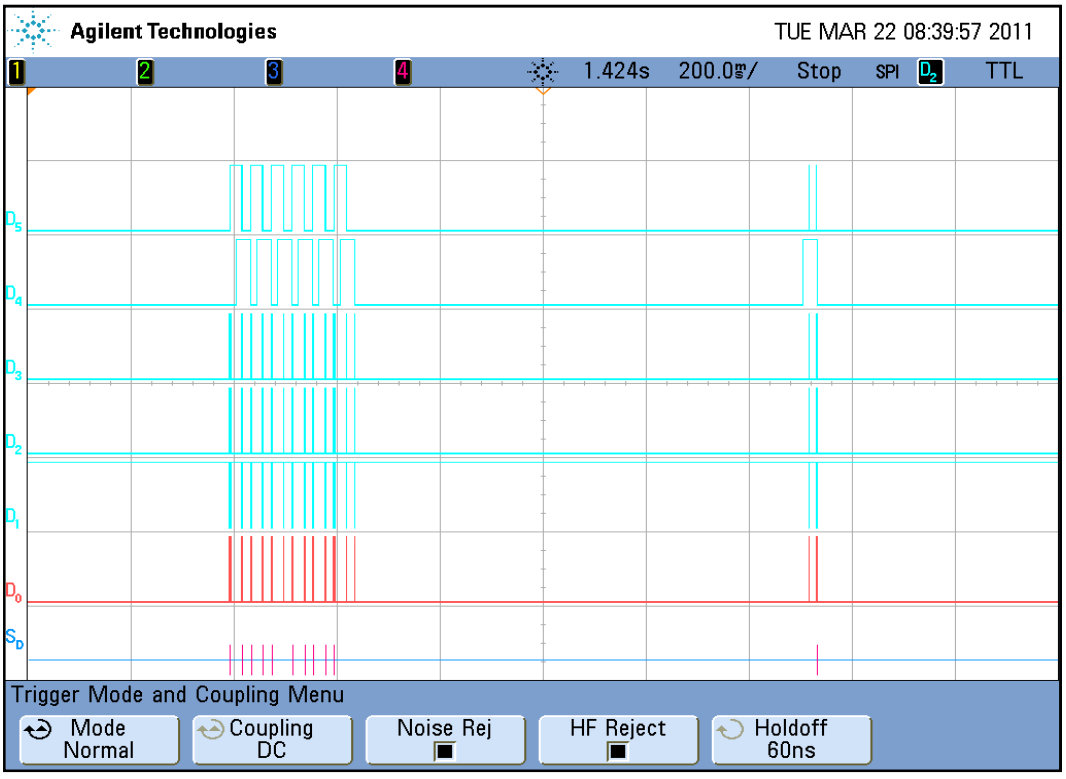


Figura 6.24 Envío de 6 paquetes y recepción de uno.

6.2 Análisis de resultados

6.2.1 Análisis de resultados de las mediciones de Radiofrecuencia

Para la figura 6.1 se puede hacer una comparación con el valor teórico del ancho de banda para una señal FSK, conociendo que el baudrate es 38,4kbuad y que la desviación de frecuencia es de 31,7383kHz se puede hacer un estimado teórico del ancho de banda esperado. Según la teoría planteada en el capítulo 3, si se cumple la condición de la ecuación 3.6, el ancho de banda para una señal FSK se puede estimar según la ecuación 3.7.

Entonces para una señal FSK con una velocidad de símbolos de 38,4kbauds, el ancho de banda BW, es:

$$BW = 153.6kHz \tag{6.1}$$

Si se compara el valor anterior con el medido en la figura 6.1, BW=130.13kHz, se tiene un porcentaje de error de 15.3%, lo cual no es aceptable. Sin embargo hay que considerar que la condición en 3.7 no se cumple y que se tiene la condición:

$$f_H - f_L < 2f_b \tag{6.2}$$

Entonces bajo la condición 6.2 se puede estimar que el BW de la señal real si es un poco menor que el estimado por 6.1. Por ende, se acepta el valor en la figura 6.1 como valor válido. Este valor es utilizado para estimar el ancho de banda del filtro de entrada, bajo la condición de que el BW en la figura 6.1 debe ser el 80% del ancho de banda filtro de entrada (BWf). Por ende:

$$BWf \cong 162.5\ [kHz] \tag{6.3}$$

Para cumplir con 6.3 se deben programa los registros MDMCFG4.CHANBW_E y MDMCFG4.CHANBW_M según la tabla 5.5.

Para las tablas 6.1 y 6.2 sólo se comentará que los porcentajes de error son bastante aceptables, pero que deben ser así debido a que la desviación de frecuencia es bastante pequeña con respecto a la frecuencia central, o dicho de otra forma la variación es mínima en comparación con la referencia y por eso ésta debe ser muy precisa.

Las mediciones de potencia se hacen como verificación de la programación del CC1101 y no se pretendió nunca realizar optimizaciones en ese sentido ya que el CC1101 se comportaba de buena forma a este nivel de potencia por lo que se decidió seguir las recomendaciones de los otros miembros del equipo. La pérdida en la potencia en la figura 6.2, puede ser más una atenuación debida a los adaptadores usados para acoplar el conector SMA a la entrada del equipo. Lo importante es revisar en la figura 6.3 que la potencia con la que se estaba recibiendo es mayor a la potencia de sensibilidad del CC1101, que llega a -112dBm a 1.2kBaud.

6.2.2 Análisis de los resultados en el funcionamiento.

Con respecto a las mediciones de la tabla 6.3 se presentan los valores medidos de manera digital, esto quiere decir que la velocidad de transmisión se estima con base en la medición del tiempo en el que se dan los eventos de inicio y fin de transmisión. Conociendo la velocidad a la que se transmiten un grupo de bytes es posible hacer un estimado del tiempo que se tarda par la transmisión de cada byte. Los valores esperados en la tabla se calculan con base en el valor programado para el baud rate. Como se menciona en el marco en el capítulo 3, existe una relación entre la velocidad de los símbolos y la velocidad de transmisión de 1:2 debido a que la codificación Manchester requiere de la transmisión de dos símbolos para representar un bit.

Lo anterior quiere decir que para una velocidad de símbolos de 38.4kbauds se tiene un bit rate de 19.2kbps, esta velocidad podría expresarse también como 2,4kBps (en bytes). Esto corresponde a un tiempo de 416.7 μs por cada byte transmitido, por otro lado se mide

un tiempo de 4.65 para la transmisión de 11 bytes, lo que sugiere que se tarden 422.7 µs por cada byte. Al final se obtiene un porcentaje de error aceptable ya que no supera el 5%.

La siguiente sección es un poco más digital, se realizaron mediciones con el analizador digital para confirmar el funcionamiento adecuado de algunas de las funciones del Ethmesh. En la figura 6.4 se ilustra el resultado para la carga de los registros de configuración en el Ethmesh, puede notarse en la figura que se envía previamente un comando para enviar al estado de espera (ver tabla 5.6), esto se hace de esa forma ya que según la hoja de datos es recomendable, más no necesario, que el CC1101 se encuentre en estado de espera para realizar configuraciones. Otro detalle es ver la diferencia entre los tipos de acceso, en la figura 6.4 se utiliza un acceso simple y por ende puede verse como antes de cada valor de configuración se envía un número de registro. Esta técnica tiene la ventaja de que se puede acceder a registros del CC1101 de manera alterna, ya que como se visualiza en la figura 6.5 el método de acceso burst requiere que los registros sean consecutivos. Pueden compararse los valores en la tabla de A.4 con los enviados en las figuras 6.5 y 6.6.

Las siguientes pruebas realizadas fueron de conectividad del Ethmesh sobre la red LAN, para ello se configura el enrutador de forma que reparta direcciones IP a partir de la dirección número 192.168.1.100 (en B.6 se muestra una imagen), de esta manera el Ethmesh conserva su dirección IP (192.168.1.50), mientras que las demás computadoras reciben una dirección asignada por DHCP. Una vez montado el Ethmesh en la red LAN se prueba la conectividad desde dos computadoras a través de un mensaje ICMP de ping. En A.6 se muestra el resultado desde ambas computadoras luego de haber hecho el “ping” al enrutador y posteriormente al Ethmesh.

Luego de confirmar la conectividad, se realiza un envío desde una PC al nodo Ethmesh. En esta prueba puede verificarse la recepción de paquetes UDP del Ethmesh, en la figura 6.7 se visualizan dos pantallas en una correspondientes a las pruebas realizadas

con la herramienta WireShark. En la parte superior de la figura pueden verse todas las configuraciones realizadas, además si se revisa la primero parte del mensaje en la ventana inferior puede distinguirse el mensaje "CONFI" enviado en los primeros bytes. Este mensaje es para que el administrados de mensajes de configuración del puerto 6651 pueda distinguir la tarea que se está solicitando. Como puede verse en la figura 6.7, efectivamente el Ethmesh interpreta la instrucción correctamente ya que responde con un mensaje "Configuración del Nodo finalizada". Esta metodología de mensajes de respuesta fue de gran utilidad para verificar el comportamiento adecuado del Ethmesh, y a la vez brindará al usuario realimentación acerca de las operaciones realizadas. En B.8 se presenta la forma de la plantilla utilizada en este caso para realizar las configuraciones, sin embargo es importante resaltar en este punto que esta es una facilidad que se brinda al usuario como solución temporal, ya que el Ethmesh tiene la capacidad de realizar estas configuraciones si se recibe la instrucción desde cualquier otro software, claro está siempre y cuando se mantenga la forma del paquete de la figura 6.7. Nótoese que luego de la frase CONFI, comienzan a enviarse las direcciones de registro y las configuraciones para cada uno.

En 6.8 se visualiza la carga de las configuraciones por defecto para el Ethmesh, estas configuraciones se encuentran escritas dentro del Ethmesh y por ende no es necesario su envío por UDP, a diferencia del caso anterior que requería de valores personalizados para cada registro. Véase de la figura 6.8 que únicamente se envía el mensaje "AcuMine" dentro del paquete de solicitud y que está dirigido por el puerto de configuraciones 6651. En la parte inferior de la figura se muestran los datos cargados por el Ethmesh en su etapa de radio, de forma que el usuario puede verificar lo que ha cargado y predecir el comportamiento de la etapa de radio. Los valores por defecto del Ethmesh están programados de manera compatible con la tecnología Acumine, así que basta con realizar la configuración inicial, las configuraciones personalizadas son una flexibilidad que se le da al Ethmesh.

Existe una opción más en la configuración del Ethmesh que tal como se describió en el capítulo 5, se utiliza para configurar el tamaño del paquete y la ubicación de la dirección de nodo. En la figura 6.9 se observa cómo se envía dicha configuración desde la PC utilizado un paquete que contiene un sencillo mensaje "PS". Así mismo en B.10 la selección realizada para dicha prueba. Es importante notar que se recibe un mensaje de confirma después de realizada la configuración. La programación del tamaño del paquete es clave porque permite al Ethmesh modificar el manejo de las capturas y envíos de paquetes hacia la red de radiofrecuencia, si un tamaño está mal dimensionado puede provocar pérdida en los bytes correspondientes a un paquete o por el contrario exceso de datos, que no sería tan crítico en recepciones, porque simplemente se descartan. Pero en envíos por la banda de 433MHz si es crítico porque el envío de bytes extras produce recarga innecesaria sobre la red.

También se hicieron pruebas para verificar la lectura de los registros de estatus del CC1101 desde una PC, en la figura 6.10 se visualiza el caso de una lectura del estado de la máquina de control del transductor. Nótese que la respuesta recibida por parte del Ethmesh es que el estado del CC1101 es 0x0D, de la tabla en B.5 el estado al que este valor corresponde es recepción, lo cual es lo esperado ya que es el estado natural en el que el microcontrolador mantiene al transductor.

En la figura 6.13 se muestran los envíos desde una PC al Ethmesh para su posterior envío a un nodo en la banda de 433MHz. Es importante notar en las figuras 6.11, 6.12 y 6.13 el protocolo que se plantea para evitar la saturación del buffer interno del Ethmesh, en las figuras se nota cómo el Ethmesh contesta a una petición para aceptar un datagrama que posteriormente es enviado por radio hacia algún camión específico. Retomando, la configuración de espacio para la dirección destino, permite ajustar el lugar en el que se buscará la dirección destino con la que se encuentra asociada una dirección IP en la tabla de direcciones. En la figura 6.13 se observa que tras una aceptación desde el Ethmesh, se envían los 63 bytes a transmitir en un paquete RF. Otro detalla a resaltar es el encabezado en los primero 5 bytes del paquete, nótese que los primeros dos bytes tienen un valor 0xFF, que como se explicó en el capítulo 5 se hace para que todos los nodos acepten el paquete y

luego sea descartado por software. Seguido de los bytes 0xFF están el numero de grupo (0x08 corresponde a camiones), y tipo (0x7D es un mensaje ping). Luego viene el byte de tamaño que en este caso es 56bytes (0x38). Es necesario recordar que de los 63 bytes hay que restar los 5 del encabezado y 2 del CRC agregados al final. Un último detalle es verificar el número de puerto (6653), que debe ser el usado para los servicios de enlace.

En las figuras 6.14 a la 6.17 se comprueba la lógica para el enrutamiento de las computadoras en la estación de control, para recibir paquetes desde algún nodo RF. El envío del mensaje de solicitud de inscripción en la tabla de enrutamiento se hace por el puerto 6653 tal y como se muestra en 6.14, el tiempo en que se realiza esta acción es t1 = 43.522004 s. El séptimo byte indica el tiempo del enlace (TTL), en este caso se deberían recibir paquetes durante 2 minutos. Luego en la figura 6.15 se visualiza la respuesta desde el Ethmesh una vez hecha la solicitud, en este caso el nodo de enlace contesta con el mensaje de confirma por el puerto 6653, tal y como era lo esperado.

En la figura 6.16 se nota cómo en un tiempo 44.538322 s, se recibe el primer paquete por el puerto 6652, que tal y como se discutió es el puerto por el cual las PC de en la estación central realizan la escucha. Se puede comparar el encabezado recibido con el encabezado mencionado cuando se analizó la figura 6.13, además véase que el tamaño del paquete sigue siendo el mismo, y corresponde al tamaño de paquetes para mensajes ping entre los vehículos de la mina. El hecho de que este paquete pase por la interfaz de Ethernet hasta la PC, significa que es un paquete válido que ha pasado por la prueba del CRC, revisada internamente en el Ethmesh.

La recepción continúa hasta un tiempo t2 = 112.533873 s. Si se hace la resta t2-t1, se obtiene un valor de 69.011869, pero se esperaba un valor de 2 minutos. El tema aquí es que la solicitud entró aproximadamente nueve segundos de que se venciera el intervalo de un minuto contado por los temporizadores 4 y 5. Esta imprecisión ocurrirá siempre que el momento de la inscripción sea muy próximo a la generación de la interrupción sin

embargo, nunca superará un minuto, quiere decir que el margen de error en el ajuste del TTL será ±1 min. Pero como para realizar el control de las descargas y alarmas se requieren tiempos mucho mayores a un minuto, el error es aceptable.

La figura 6.18 es un ejemplo de cómo las herramientas del MPLAB también facilitaron comprobar algunos aspectos de la programación del Ethmesh. En este se muestra como se utilizó el "watch" y un punto de parada para visualizar el primer valor de una IP inscrita en la tabla de enrutamiento. Para esa solicitud en especial, se ajusta un TTL de 11 minutos y una máscara de RF de 12.

En las figuras 6.19 hasta la 6.24 se comprueban dos cosas básicas, por un lado la capacidad del Ethmesh de captar los paquetes desde la banda de 433MHz y por otro lado que los paquetes recibidos por UDP son enviados al CC1101 para que éste realice las transmisiones en forma ordenada y asistida por el microcontrolador.

En la figura 6.19 se visualizan las dos interrupciones de umbral en la línea D5 y la interrupción de fin de paquete en D4. Nótese que D4, correspondiente a GDO2, se levanta antes que D5 (GDO0) y esto es lo esperado ya que cuando esto ocurre es porque se ha captado alguna palabra de sincronización y el demodulador del CC1101 está comenzando a recibir los paquetes y empujándolos en la FIFO_RX. En 6.20 se realiza un acercamiento para comprobar que la lectura de la FIFO_RX da inicio, las líneas D3 y D2 corresponden a SDI y SDO (desde el punto de vista del PIC) y la línea D1 es el CSn, el equipo ha sido configurado para decodificar la línea de lectura desde la FIFO_RX (SDI). Se ve aparecer de nuevo el encabezado 0xFF, 0xFF, 0x08 y 0x7D en la primera lectura desde la FIFO_RX. En la figura 6.21 aparecen las lecturas realizadas para el segundo evento de umbral.

En la figura 6.22 se comprueban los envíos por RF al verificar las escrituras a la FIFO_TX. El orden de las líneas es el mismo (D1 es CSn, D3 es SDI, D2 es SDO y D0 es SCLK), se pueden notar que para escribir un paquete de 67 bytes se escriben 63 bytes al FIFO_TX, cuando se sobrepasan los 33bytes la línea D5, correspondiente a GDO0, se levanta el nivel de umbral pero para transmisión el flanco en interés es el negativo ya que este indica el momento de la recarga en FIFO_TX. En 6.22 se visualizan dos envíos completos y parte de otro. En la figura 6.23 se hace una ampliación en la escala que permite verificar la escritura de los últimos 10 de 63 bytes, nótese que una vez terminado el llenado de FIFO_TX, se envía un comando para mover al transductor al estado de transmisión. Esta acción es la esperada, ya que en el momento que se escriben los bytes, el CC1101 se encuentra en RX y por ende es necesario un comando para que entre en modo transmisión.

Finalmente en la figura 6.24 se realiza una comprobación muy importante, ya que allí se visualizan las dos operaciones requeridas para la comunicación del Ethmesh con los vehículos. Primero se puede distinguir cómo se envían 6 paquetes al Ethmesh, una vez que éste terminó de enviarlos se pasa al estado de recepción de nuevo, más a la derecha puede verse que el Ethmesh recupera su estado natural y es capaz de recibir un paquete una vez completada la transmisión.

Capítulo 7: Conclusiones y recomendaciones

7.1 Conclusiones

1. El nodo Ethmesh presta un servicio de enlace que consiste en colocar los paquetes de un lado al otro de la red, pero no tiene la capacidad de dar seguimiento de los paquetes ni de confirmar las entregas.

2. Para la entrega de un paquete a un nodo RF destino es necesaria la asistencia de un nodo administrador ya que el Ehtmesh no tiene la capacidad de comportarse como un sumidero en la red inalámbrica.

3. El Ethmesh puede ser configurado para buscar en distintas posiciones del paquete una dirección de nodo destino, esto le da flexibilidad para el enrutamiento de mensajes RF a una red LAN.

4. El inicio de una transmisión RF en el Ethmesh está limitada únicamente a no encontrarse en medio de un proceso de recepción de un paquete y por ende, puede darse casos de colisiones en el canal de 433MHz.

5. Las desviaciones en la frecuencia portadora son aceptables por lo que no es necesario realizar una compensación en la desviación de la frecuencia.

6. El efecto de embotellamiento de los datos, provocado por la diferencia de velocidades en ambas redes, puede ser manejado con técnicas de almacenaje temporal de los datos y un control de flujo en la entrada de alta velocidad.

7. El manejo de paquetes de tamaño variable depende del establecimiento previo del tamaño de los paquetes RF para poder manejar de manera coherente los eventos de recepción y transmisión.

8. A pesar de que el Ethmesh puede configurarse remotamente para operar a distintas frecuencias, existe una limitante en los componentes electrónicos con los que se construyó la etapa de radiofrecuencia que no garantiza el buen funcionamiento en frecuencias no contenidas en el rango de 315-433MHz.

7.2 Recomendaciones

1. Se recomienda realizar un estudio de manera que se puedan utilizar las funciones de evaluación de canal libre incorporadas en el transductor, para evitar colisiones y así mejorar el uso del canal.

2. Para aplicaciones en las que el consumo de potencia es un factor crítico, sería conveniente utilizar un manejo de la potencia con una intensidad variable, esta función también está disponible para el transductor utilizado sin embargo, es necesario realizar una programación especial de los registros “PTABLE”.

3. Para optimizar las operaciones de transmisión puede hacerse uso del indicador de calidad de enlace incorporado en el CC1101, de manera que se evalúa la condición del enlace antes de realizar una transmisión por ende, previene que un nodo desperdicie potencia en un intento por transmitir bajo condiciones no óptimas en el canal.

4. Para aplicaciones de distancias variables que incluyen distancias muy cercanas y lejanas es necesario bajar la potencia para transmisiones de corta distancia y así no saturar al receptor. Una manera de hacer esto es transmitir a distintos niveles hasta recibir un reconocimiento.

5. Queda pendiente programar de manera correcta los lazos de ganancia para la medición del RSSI en el CC1101 y utilizar este parámetro como referencia para evaluar la ocupación del canal.

6. Es posible utilizar un software en las computadoras que sea más flexible y tenga una interfaz más cómoda para el usuario de la estación central de control.

Referencias

[1] "Instituto de Investigaciones en Ingeniería Eléctrica". El contenido de esta página web justifica la existencia de un ente de investigación así como las distintas áreas de trabajo y todo lo referente al personal y la historia del instituto. Tomado de: http://www.iiie-conicet.gov.ar. El 11 de abril de 2011.

[2] Nota de aplicación "315-433MHz Reference Design (swrr046.zip)", fabricante SMSC. Tomado de: http: //focus.ti.com/docs/toolsw/folders/print/cc1101em433_refdes.html. El 2 de setiembre de 2010.

[3] Autoxuga móvil, artículo "Electrónica Digital y Can Bus". Tomado de: http://www.autoxuga.com/cursos/ELECTRONICA/ELECTRONICA.htm, el 20 de agosto de 2010.

[4] Hoja de datos del CC1101, fabricante Texas Instruments. Tomado de: http: //search.digikey.com/scripts/DkSearch/dksus.dll?Cat=3539948&k=CC1101, el 20 de agosto de 2010.

[5] Nota de aplicación AN001 "SRD Regulations for Licence Free Transceiver Operation", fabricante SMSC. Tomado de: http://focus.ti.com/docs/toolsw/folders/print/cc1101em433_refdes.html, el 2 de setiembre de 2010.

[6] Armano, Sebastian, "Nodos para Redes de Sensores en 433 MHz" Directores: Favio Masson, Pablo Mandolesi. Biblioteca Universidad Nacional del Sur.

[7] Retana Durán, Elías, "Prácticas de Modulación Digital con Equipo de Radiofrecuencia" Directores: M.Sc Victor Hugo Cahcón Prendas. Tomado de: http://eie.ucr.ac.cr/uploads/file/proybach/pb0547t.pdf

[8] Schilling, Donald L. Taub, Herbert. "Principles of Comunication Systems", Second Edition. McGraw-Hill Publishing Company.

[9] L. Aaron Kaplan. "Funkfeuer.at & community wireless networks running MANET protocols". Presentación pdf, tomado de:

[10] http: //wiki.tools.ietf.org/agenda/80/slides/manet-5.pdf, el 5 de mayo de 2011.

[11] Redes Ad hoc, Wikipedia. Tomado de: http: //es.wikipedia.org/wiki/Red_Ad_hoc, el 28 de abril 2011.

[12] Wikipedia, "Distance-vector routing protocol", Tomado de: http: //en.wikipedia.org/wiki/Distance-vector_routing_protocol, tomado el 15 de abril 2011.

[13] Wikipedia, "Ad hoc On-Demand Distance Vector Routing". Tomado de: http: //en.wikipedia.org/wiki/AODV, tomado el 17 de abril de 2011.

[14] C. E. Perkins, E. M. Royer, "Ad Hoc On-Demand Distance Vector Routing", Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100, February 1999.

[15] Universidad Rey Juan Carlos, "Encadenamiento en redes ad hoc". Tomado de: http: //gsyc.es/~mortuno/rom/05-encaminamiento_adhoc.pdf, el 12 de abril 2011.

[16] Boix Requesens,Antoni, "diseño e implementación de un protocolo de transporte para una red ieee 802.15.4 con protocolo de encaminamiento nst-protocolo de encaminamiento, nst-aodv". Tomado de: http://upcommons.upc.edu/pfc/bitstream/2099.1/8776/1/PFC_CD.pdf, el 12 de abril 2011.

[17] Poor, Robert, "Wireless Mesh Networks". Tomado de: http://www.sensorsmag.com/networking-communications/standards-protocols/wireless-mesh-networks-968 el 12 de abril de 2011.

[18] Baldomero, Coll y Gozálvez, Javier "Comunicaciones Multi-hop en Redes Mesh Inalámbricas". Tomado de: http://www.uwicore.umh.es/files/paper/2009_national/uwicore_ursi2009_Mecanismos%20de%20Seleccion%20de%20Vecinos%20para%20Comunicaciones%20Multi-Hop%20en%20Redes%20Mesh%20Inalambricas.pdf, el 20 de abril de 2011.

[19]] T. Clausen. P. Jacquet. "Request for Comments: 3626". Tomado de: http: //www.ietf.org/rfc/rfc3626.txt, el 21 de abril 2011

[20] Wikipedia, "Optimized Link State Routing". Tomado de: http://es.wikipedia.org/wiki/Optimized_Link_State_Routing, el 25 de abril 2011

[21] P. Jacquet & P. Jacquet, "Request for Comments: 3626, Optimized Link State Routing Protocol". Tomado de: http://www.ietf.org/rfc/rfc3626.txt, el 25 de abril 2011

[22] D

[23] Buettrich, Sebastian, "Redes Mesh". Tomado de: http: //www.it46.se/courses/wireless/materials/es/13_Redes-Mesh/13_es_redes_mesh_presentacion_v01.pdf, visitado el 8 de setiembre 2010.

[24] Microchip publicaciones de soporte "Ethernet solutions with integrated MAC and PHY". Tomado de: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nod|

[25] Manual de usuario y aplicación SmartRF®Studio, fabricante SMSC. Tomado de: http: //focus.ti.com/docs/toolsw/folders/print/smartrftm-studio.html, el 2 de setiembre de 2010.

[26]] Nota de aplicación “315-433MHz Reference Design (swrr046.zip)”, fabricante SMSC. Tomado de: http: //focus.ti.com/docs/toolsw/folders/print/cc1101em433_refdes.html, el 2 de setiembre de 2010.

[27] Interiano E, Montes de oca F. "TCP/IP". Tomado de: http://www.ie.itcr.ac.cr/faustino/Redes/Clase2/1.2ModeloTCPIP.pdf, visitado el 22 abril 2011.

[28] Rodríguez L. "Modelo de Referencia OSI". Tomadp de: http://www.ie.itcr.ac.cr/faustino/Redes/Clase1/ModelodeReferenciaOSI.pdf, visitado el 22 mayo 2011.

[29] IEEE Computer Society, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", tomado de: http://standards.ieee.org/findstds/errata/802.3-2008_Cor1.pdf, visitado el 22 de mayo 2011.

[30]] J. Postel, "Request for Comments: 792, INTERNET CONTROL MESSAGE PROTOCOL" tomado de: http://www.ietf.org/rfc/rfc0792.txt?number=792, visitado el 21 de octubre 2010.

[31]] J. Ortega Lopera, "Redes de altas prestaciones y sus aplicaciones", Lección 3. Tomado de: ttp://www.google.com/#sclient=psy&hl=es&source=hp&q=REDES+DE+ALTAS+PRESTACIONES+Y+SUS+APLICACIONES+leccion+3&aq=f&aqi=&aql=&oq=&psj=1&bav=on.2,or.r_gc.r_pw.&fp=472e648603ef90d6, visitado el 30 octubre 2010.

[32] Wikipedia, "Berkley Sockets". Tomado de: http://en.wikipedia.org/wiki/Berkeley_sockets#cite_note-2, visitado el 22 de mayo de 2011.

[33]] B. Hall, "Beej's Guide to Network Programming, Using Internet Sockets". Tomado de: http://beej.us/guide/bgnet/output/html/multipage/index.html, visitado el 25 de setiembre 2010.

[34] A. Rafiq, "Microchip TCP/IP Stack with BSD Socket API", Microchip Technology Inc. Tomado de:http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2615&dDocName=en545713, vistado el 20 de agosto 2010.

[35] J. Espinoza, J.Cano, A. Sepúlveda, "Redes de datos". Tomado de: http://juandeg.tripod.com/rs232.htm, visitado el 22 mayo 2011.

[36]] Information Sciences Institute University of Southern California, RFC0793 "TRANSMISSION CONTROL PROTOCOL". Tomado de: http://tools.ietf.org/html/rfc0793, visitado el 25 de setiembre 2011.

[37]] J. Postel,RFC 768 "PROTOCOLO DE DATAGRAMAS DE USUARIO", Traducción al castellano:Domingo Sánchez Ruiz. Tomado de:http://www.rfc-es.org/rfc/rfc0768-es.txt, visitado el 25 de setiembre 2011.

[38] Organización IANA, "Aplication for user port number". Tomado de: http://www.iana.org/assignments/port-numbers, visitado el 15 setiembre 2010.

[39]] Texas Instrumentes, "Errata notes CC1101". Tomado de http://focus.ti.com/lit/er/swrz020b/swrz020b.pdf, visitado el 19 enero 2011.

[40] O

[41] Hoja de datos del CC1100, fabricante Texas Instruments. Tomado de: http://search.digikey.com/scripts/DkSearch/dksus.dll?Cat=3539948&k=CC1101, el 20 de agosto de 2010.

[42] Hoja de datos del PIC32MX5XX/6XX/7XX, fabricante Microchip. Tomado de: http://www.microchip.com/en_US/family/32bit/, visitado el 20 de agosto de 2010.

[43] Manual de referencia de la familia PIC32, Capitulo 15 "Input Capture",fabricante Miccrochip. Tomado de: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2612, visitado el 5 de febrero del 2011.

[44]] Manual de referencia de la familia PIC32, Capitulo 35 "Ethernet Controller",fabricante Miccrochip. Tomado de: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2612, visitado el 5 de febrero del 2011.

Apéndices

Apéndice A.1 Imagen de Caja de adquisición, comunicación e interfaz Acumine.

Apéndice A.2 Imagen del nodo liviano.

Apéndice A.3 Protocolo de mensajes de control internet (ICMP)

El ICMP involucra un protocolo a nivel de capa de red que aparte del encabezado convencional de IP, agrega sus propios headers para identificar el tipo de mensaje que se requiere enviar y recibir. Para cualquier ICMP, aparecen ARP de por medio en caso de que no se tenga la información del IP relacionada con la MAC, de manera que se realiza una multidifusión en MAC para averiguar quién es la dirección MAC correspondiente a una IP. Luego de resolver el ARP, continúa con el envío de los paquetes ICMP. El ping y el trace-out involucran ICMP directamente y son herramientas de prueba que usan UDP por conveniencia.

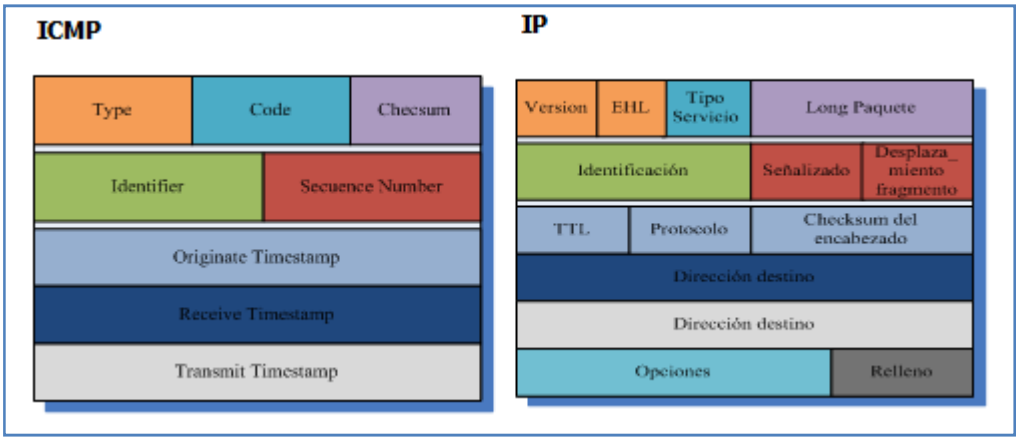


Figura A.1 Estructura de paquete ICMP e IP.

En la figura anterior se muestran las estructuras de un paquete IP y un paquete ICMP. Como puede verse el paquete ICMP es mucho más simple, este tipo es usado para mensajes de control muy útiles para evaluar el estado de una conexión simple. Pr otro lado el Traceout puede utilizarse para evaluar la ruta de un paquete y detectar el origen de los errores en una conexión. Ping envía solicitudes de respuesta para probar conectividad IP entre host, utiliza el protocolo de capa 3 ICMP y mide el tiempo de respuesta, el valor de dicho tiempo puede ser limitado. Por ejemplo ping con dirección 127.0.0.1 prueba la configuración del host local. ICMP es capaz de generar notificaciones que son visibles por el administrador de una red. Las notificaciones se pueden hacer por host inalcanzable, red inalcanzable, protocolo inalcanzable, puerto inalcanzable, tiempo de vida superado, redirección de rutas y disminución de la velocidad [29].

Apéndice A.4 Tabla con las configuraciones de los registros del CC1101.

clasificación	Dirección	Registro	Valor Hexa	Comentario
Control de la máquina de estados	0x00	iocfg2	6	valor por defecto (Chip ready signal)
Control de la máquina de estados	0x01	iocfg1	2E	reservado para SDO, tercer estado
Control de la máquina de estados	0x02	iocfg0	0	valor por defecto (salida del cristal entre 192)
Control de la maquina de estados	0x03	fifothr	47	mitad del buffer aproximadamente es el threshold para generar la interrupción de RXFIFO,
Sincronización y manejo de paquetes.	0x04 y 0x05	sync1 ,sync0	33 CC	palabra de sincronización 0011001111001100
Sincronización y manejo de paquetes.	0x06	pktlen	43	paquetes de 67 bytes
Sincronización y manejo de paquetes.	0x07	pktctrl1	C3	filtrado por dirección con posibilidad de broadcast 0x00 ó 0xFF, sin vacío de FIFO RX por fallo en CRC, sin añadir RSSI al payload, PQT=3
Sincronización y manejo de paquetes.	0x08	pktctrl0	0	Sin whitening, modo normal usando FIFOS para TX y RX, sin CRC y tamaño fijo.
Sincronización y manejo de paquetes.	0x09	addr	FF	dirección 255 para este nodo
Parámetros de la modulación FSK	0x0A	channr	0	Canal por default
Parámetros de la modulación FSK	0x0B	fsctrl1	6	Fif=152kHz, asumiendo cristal de 26MHz. Valor sugerido por el RFStudio
Parámetros de la modulación FSK	0x0C	fsctrl0	0	sin compensación de frecuencia

Tabla con las configuraciones de los registros del CC1101 (continuación)

Parámetros de la modulación FSK	0x0D	freq2	10	Frecuencia base para el sintetizador, o sea la portadora.
Parámetros de la modulación FSK	0x0E	freq1	A7	Portadora 433MHz
Parámetros de la modulación FSK	0x0F	freq0	60	Portadora 433MHz
Parámetros de la modulación FSK	0x10	mdmcfg4	9A	Medido y comparado con la teoría, se encontro necesaria la reprogramacion de este parametro de forma que el nuevo ancho de banda quedó en 162,5 kHz.
Parámetros de la modulación FSK	0x11	mdmcfg3	83	data rate de 38,3kbps, velocidad requerida para la sincronía con las cajas Acumine
Parámetros de la modulación FSK	0x12	mdmcfg2	A	sin DC bloking para el filtro del demodulador, 2 FSK, manchester y 16 de 16 para la palabra de sincronización.
Sincronización y manejo de paquetes.	0x013	mdmcfg1	72	sin corrección de error con interleaving para el payload, exponente de separación de canal igual a 2,y 24 bytes de preámbulo
Parámetros de la modulación FSK	0x14	mdmcfg0	93	separación de canal de 159.088 KHz, parámetro no crítico ya que se usa un canal
Parámetros de la modulación FSK	0x15	deviatn	42	Desviación de 31,7 KHZ. Parámetro utilizado por las cajas
Control de la máquina de estados	0x16	mcsm2	7	cuando el tiempo de RX termina la condición para mantenerse en estado de recepción es que haya encontrado una palabra de sincronización, el tiempo Rxtimeout es hasta terminar de recibir un paquete.

Tabla con las configuraciones de los registros del CC1101 (continuación)

Control de la máquina de estados	0x17	mcsm1	2F	luego de rx o de tx se va a idle y transmite bajo la condición de que no esté recibiendo un paquete.
Control de la máquina de estados	0x18	mcsm0	18	cristal en modo de apagado automático cuando entra en estado SLEEP o XOFF, pin radio control apagado, calibración del sintetizador cada vez que va del IDLE a TX o RX,
Parámetros de la modulación FSK	0x19	foccfg	16	Compensación de frecuencia por un valor equivalente al ancho de banda del filtro de RX /4, mismos valores para las ganancias de lazo de la compensación de frecuencia, la compensación no se detiene aún cuando el Cs esta en bajo.
Sincronización y manejo de paquetes.	0x1A	bscfg	6C	Ganancias de los filtros PI usados antes y después de los recoveries de la palabra de sincronización, determinados por el RF Studio
Sincronización y manejo de paquetes.	0x1B	agcctrl2	43	la ganancia más alta establecida no puede ser usada, valor máximo posible para LNA, el target está en 33dBm.
Sincronización y manejo de paquetes.	0x1C	agcctrl1	40	At MAGN_TARGET, la ganancia LNA decrece primero, carrier sense relativo desactivado
Sincronización y manejo de paquetes.	0x1D	agcctrl0	91	16 muestras para el filtro del canal y un valor de FILTER_LENGTH=1
Control de la máquina de estados	0x1E y 0x1F	worevt1 y worevt0	87 6B	default value
Control de la máquina de estados	0x20	worctrl	FB	máximo timeout para evento cero, habilitado la calibración del oscilador RC para el WOR, y tiempo de evento uno entre 1.33 y 1.38 ms
Parámetros de la modulación FSK	0x21	frend1	56	

Tabla con las configuraciones de los registros del CC1101 (continuación)

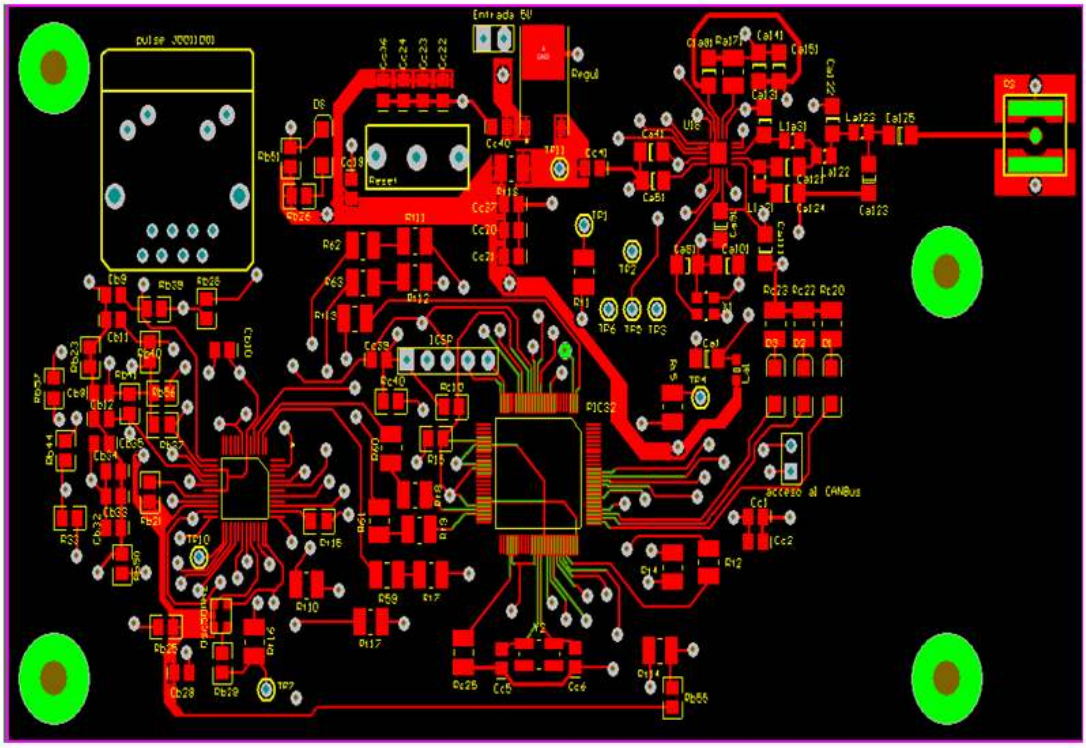
Consumo de potencia	0x22	frend0	10	sin PA ramping
Parámetros de la modulación FSK	0x23	fscal3	E9	calibración del sintetizador
Parámetros de la modulación FSK	0x24	fscal2	A	
Parámetros de la modulación FSK	0x25	fscal1	0	
Parámetros de la modulación FSK	0x26	fscal0	1F	valores del RFStudio
Control de la máquina de estados	0x27	rcctrl1	0	oscilador RC para el WOR
Control de la máquina de estados	0x28	rcctrl0	0	oscilador RC para el WOR
Prueba y debbuging	0x29	ftest	0	
Prueba y debbuging	0x2A	ptest	7F	
Prueba y debbuging	0x2B	agctest	0	
Prueba y debbuging	0x2C 0x2D 0x2E	test2, test1 y test0	81 35 9	valor de rfstudio
status	0x30	partnum		numero de parte
status	0x31	version		versión del chip
status	0x32	freqest		frecuencia estimada por el sintetizador
status	0x33	lqi		estimación del demodulador del LQI
status	0x34	rss		fortaleza de la señal recibida
status	0x35	marcstate		estado actual del control del cc1101
status	0x36	wortime1		
status	0x37	wortime2		entre este y el anterior indican el tiempo contado en estado de sleep.

Tabla con las configuraciones de los registros del CC1101 (continuación)

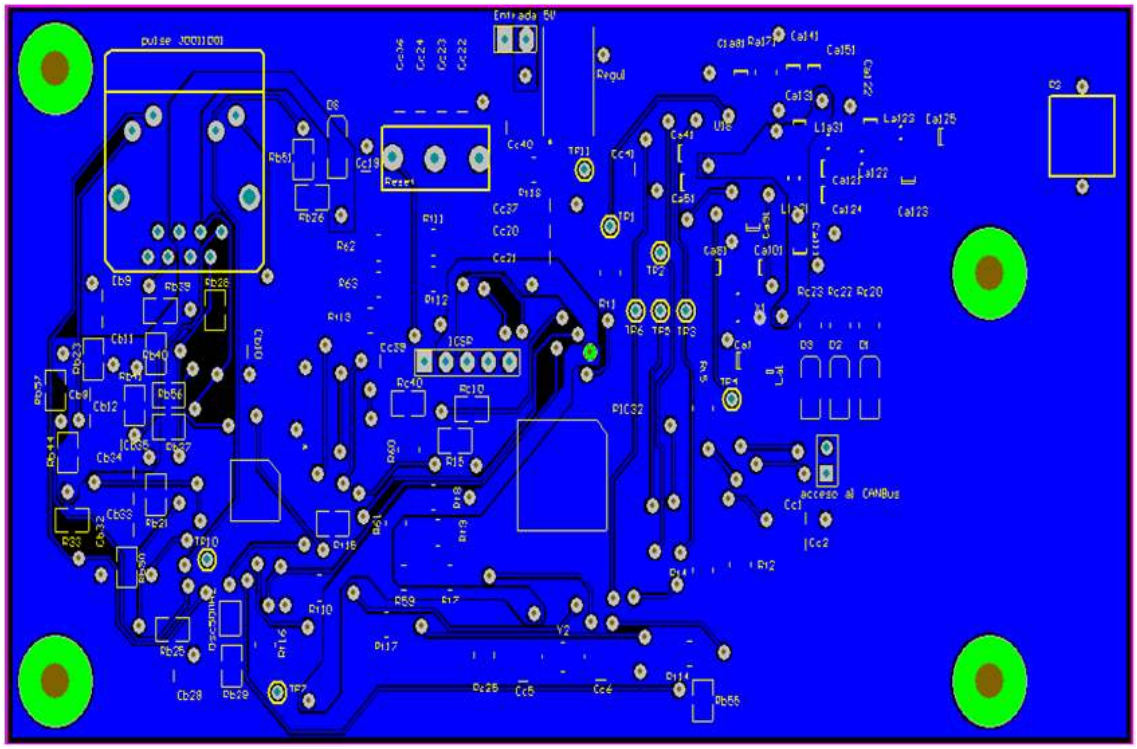
status	0x38	pktstatus		valor del GDOx actual y estado del paquete
status	0x39	vco_vc_dac		
status	0x3A	txbytes		bytes disponibles en el FIFO de transmisión
status	0x3B	rxbytes		bytes disponibles en el FIFO de recepción
status	0x3C	rccctrl1_status	0	resultado de la última calibración del Oscilador RC
status	0x3D	rccctrl0_status	0	resultado de la última calibración del Oscilador RC
Consumo de potencia	0x3E	patable0	C0	Una única configuración de potencia para transmitir con 10dBm, la máxima potencia disponible en el CC1101.
Consumo de potencia	0x3E	patable1	0	valor de rfstudio
Consumo de potencia	0x3E	patable2	0	valor de rfstudio
Consumo de potencia	0x3E	patable3	0	valor de rfstudio
Consumo de potencia	0x3E	patable4	0	valor de rfstudio
Consumo de potencia	0x3E	patable5	0	valor de rfstudio
Consumo de potencia	0x3E	patable6	0	valor de rfstudio
Consumo de potencia	0x3E	patable7	0	valor de rfstudio

Apéndice A.5 Propuesta de diseño para integración del Ethmesh en una placa. (PCB y esquemáticos)

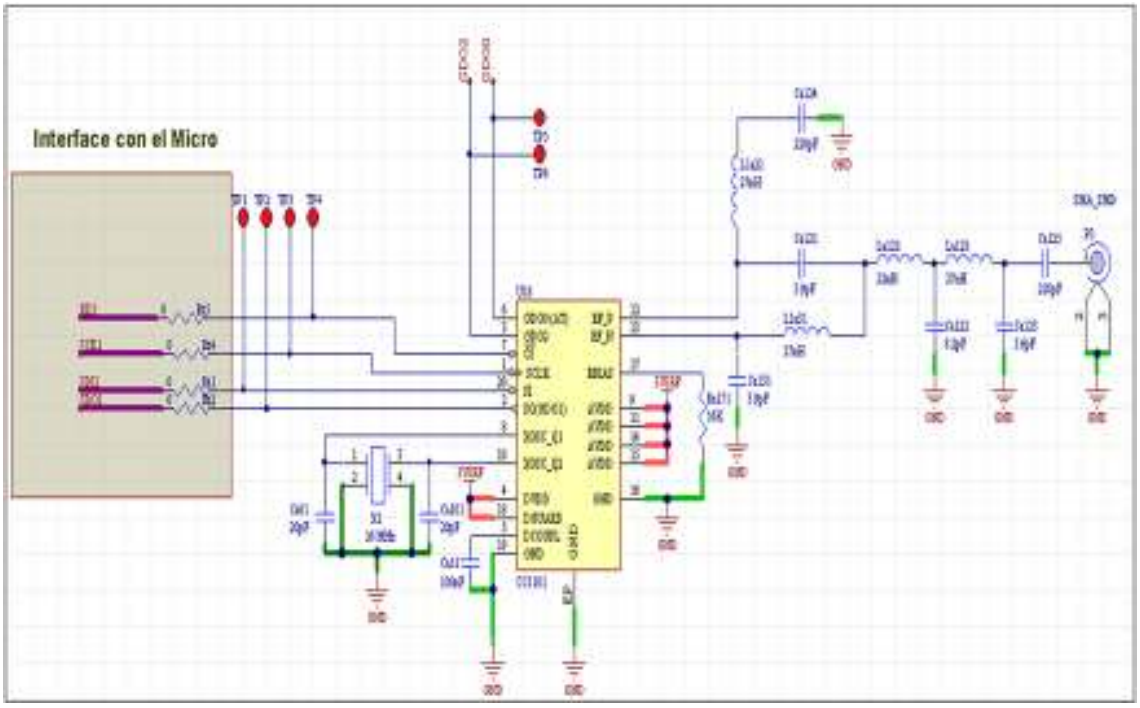
PCB, Top Layer.



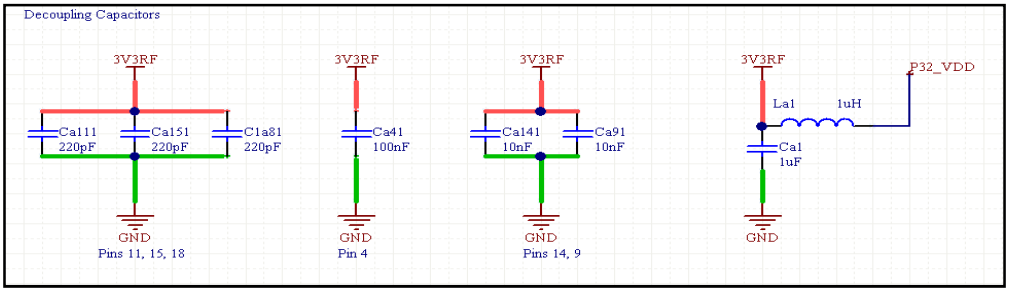
PCB, Bottom Layer.



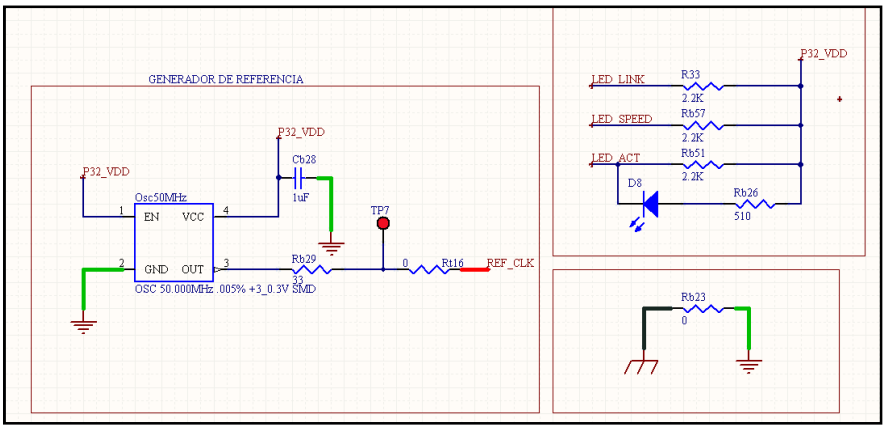
Bloque de Radio frecuencia, CC1101 y circuito balun para conexión de antena λ/8.



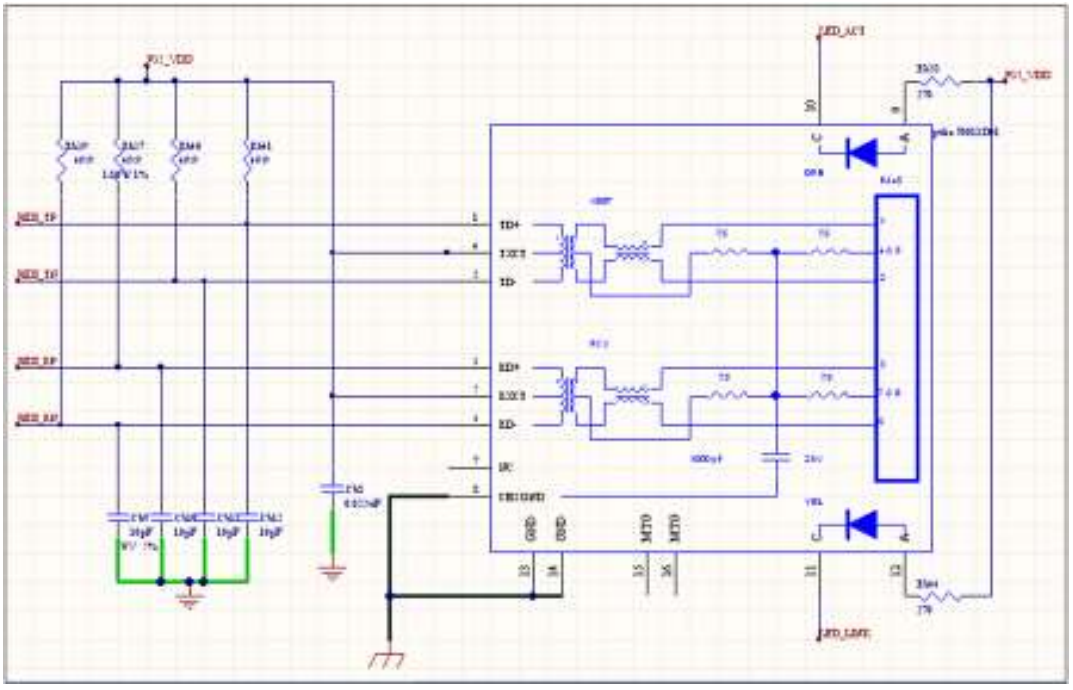
Capacitores de desacople de la etapa de RF.



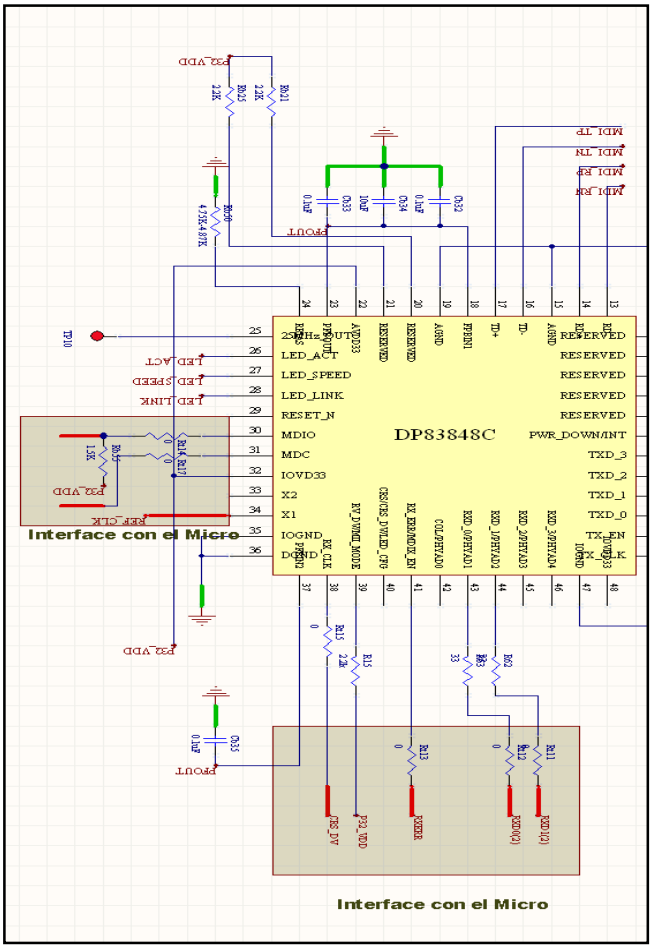
Generador de frecuencia para el transductor LAN y LED´s de indicación de conectividad Ethernet.



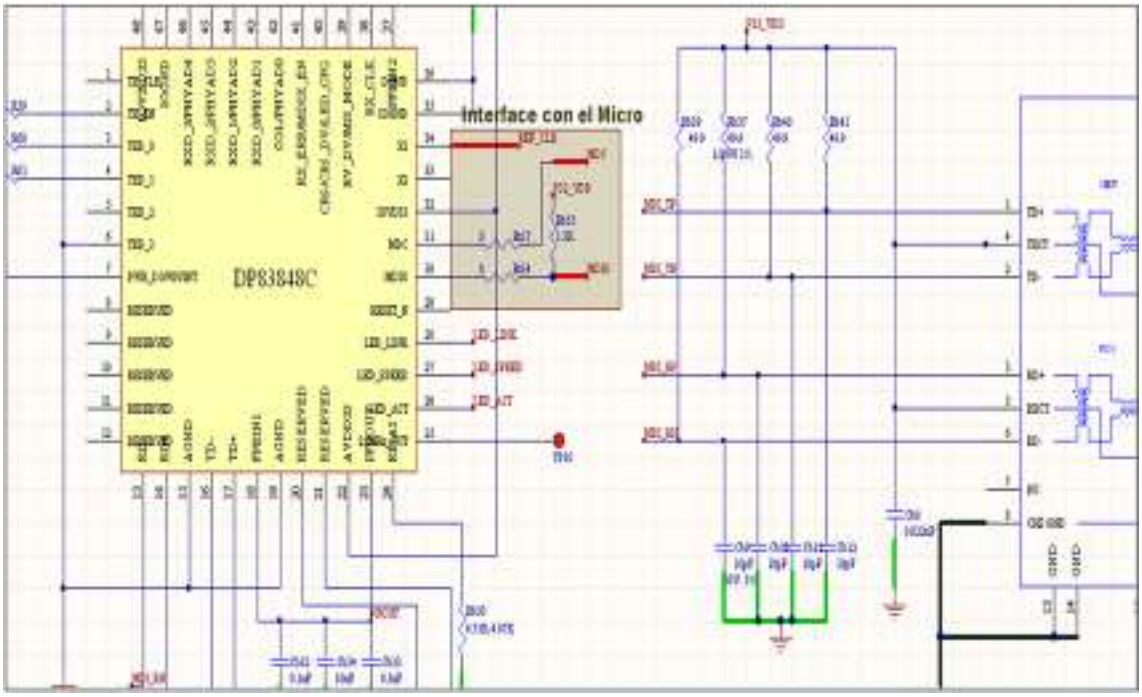
Etapa de conexión a Ethernet, conector RJ45 con acople de impedancia incluido.



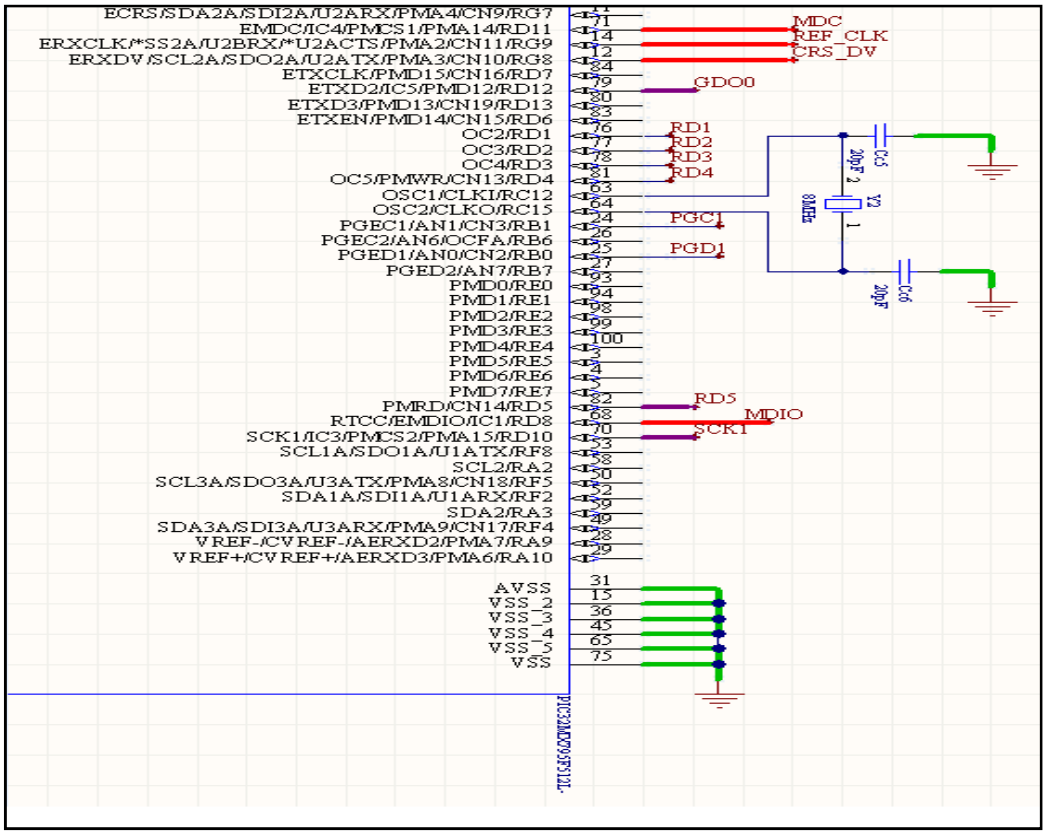
Transductor LAN y conexiones con el PIC32.



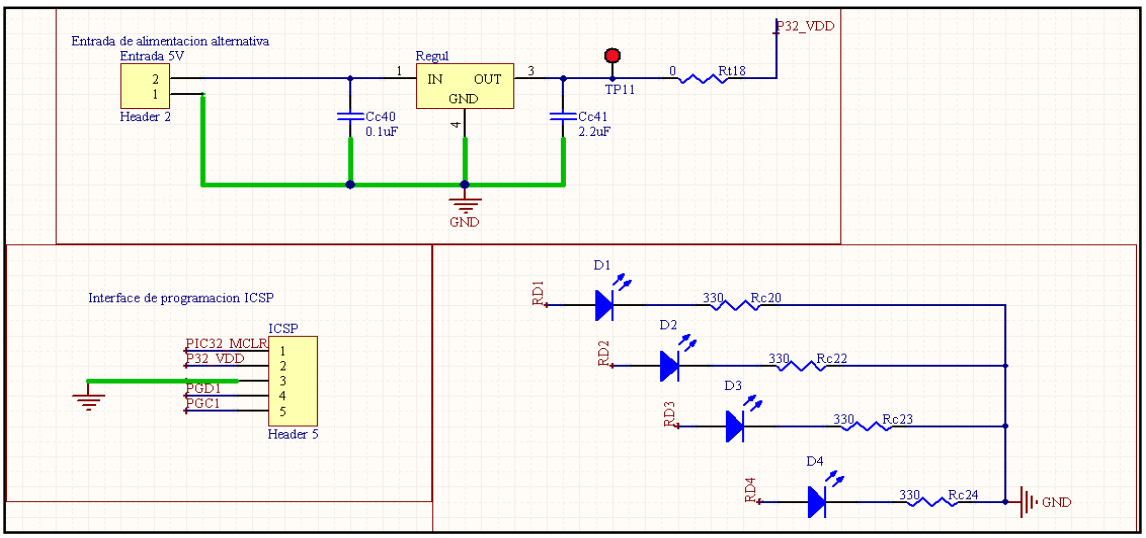
Sección faltante, interconexiones entre el DP8384C y el conector RJ45.



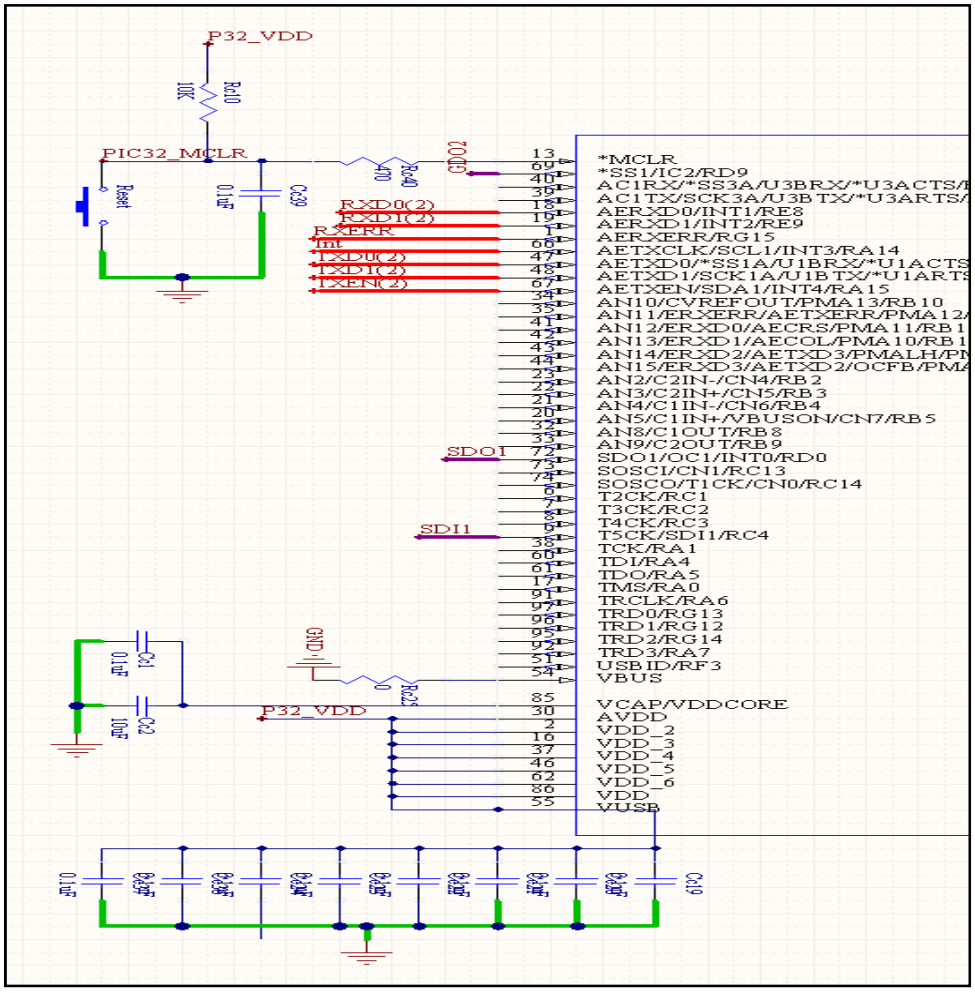
PIC32 y cristal oscilador (lado derecho).



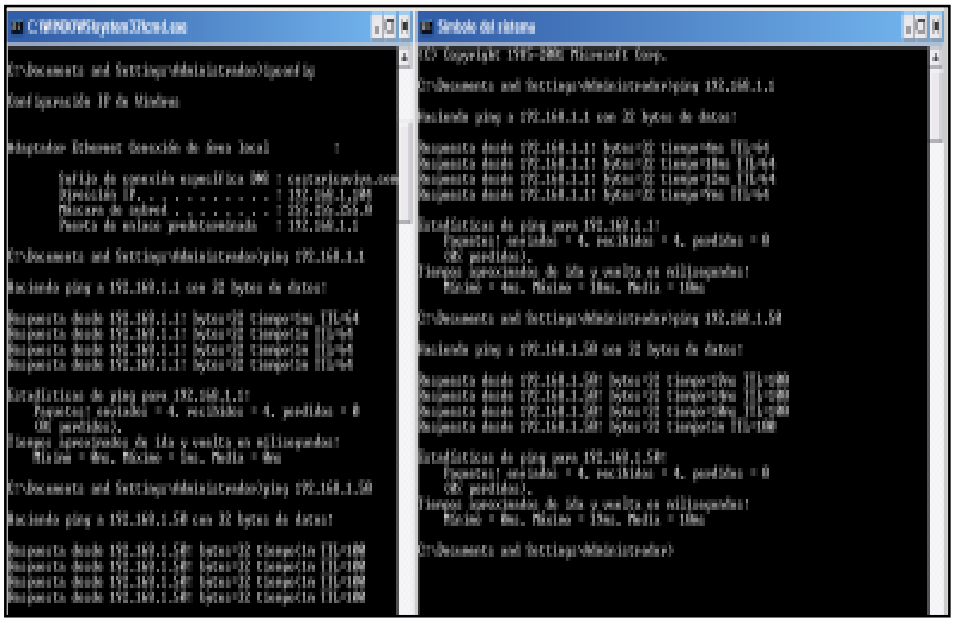
Etapas de alimentación, interfaz de programación y visualización.



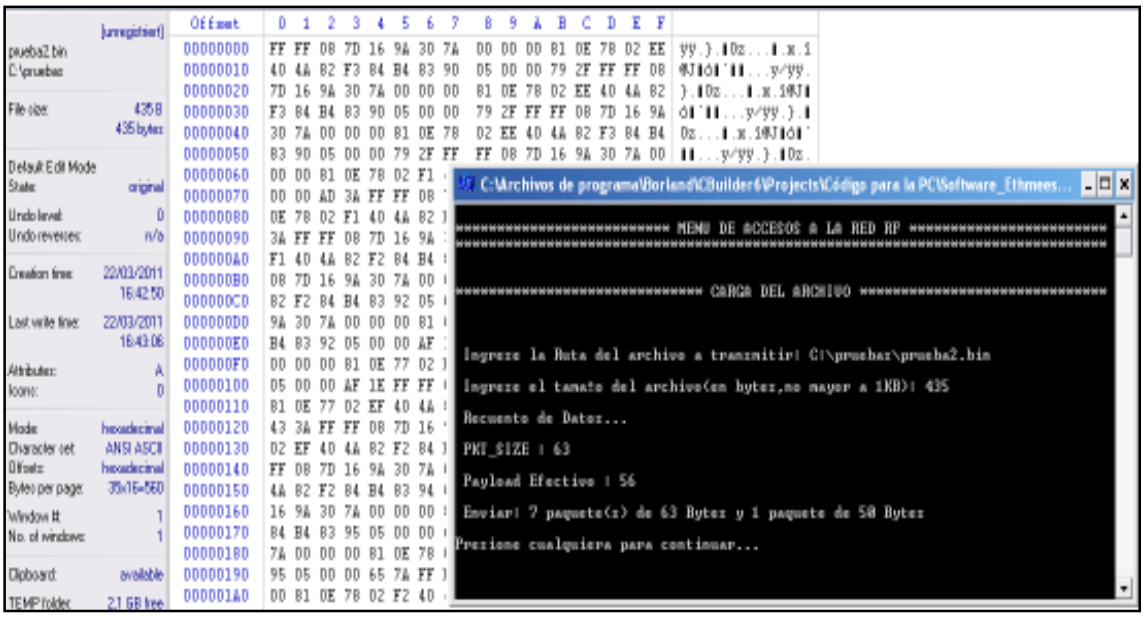
PIC32, reset y capacitores de desacople (lado izquierdo).



Apéndice A.6 Envío de mensaje ping para confirmación de conectividad



Apéndice A.7 Archivo hexadecimal segmentado para el envío en varios paquetes.



Apéndice A.8 Manual de usuario para el nodo de enrutamiento Ethmesh.

A.8.1 *Características generales y conexión del hardware*

El hardware del Ethmesh consiste en un "Ethernet Starter Kit" de Microchip y una placa de radiofrecuencia que se une al Kit por medio de un conector de 132 pines marca Hirose. Una placa metálica cubre la etapa de radio, evitando así las interferencias sobre los puntos de prueba colocados sobre la interfaz micro-transductor (ver figura A.9.1).

Figura A.8.1 Nodo Ethmesh.

Para comenzar a utilizar el nodo, éste debe conectarse al switch o enrutador que concentra la red LAN. El nodo Ethmesh tiene una dirección IP fija 192.168.1.50, para cambiarla es necesario su reprogramación, dicha reprogramación se hace cambiando el valor de la definición DEFAULT_IP_ADDR en la línea número 52 del archivo *tcp_bsd_config.h,* ubicado en la ruta ***C:\Microchip Solutions\ETHMESH_FINAL\source.*** En la carpeta Microchip Solutions, ubicada en la raíz se encuentra todo el código fuente necesario para el funcionamiento del Ethmesh. Para compilarlo y descargarlo sobre el hardware es necesaria una PC con el MPLAB_IDE versión 8.46 o mayor, además necesita el compilador C-32 para los microcontroladores PIC32. Otra forma un poco más práctica es descargar el Ethmesh.hex que se encuentra en la ruta ***C:\Microchip Solutions\ETHMESH_FINAL\Objects***, pero se esa forma no es posible que los cambios en el software se reflejen en el comportamiento del hardware, este archivo se entrega con el fin de tener un respaldo a mano en caso de una des-programación del Ethmesh.

Posteriormente coloque el dispositivo dónde sea necesario (a una distancia máxima de 50 metros de los demás nodos de la red inalámbrica), una vez alimentado el nodo un LED color verde comenzará a titilar, cada vez que esto suceda es porque se están recibiendo paquetes por el canal configurado. Al encender el Ethmesh éste realiza una configuración de la etapa de radio que es compatible con la comunicación inalámbrica de las cajas Acumine. Si se desean realizar configuraciones para la comunicación por radio, debe usar la aplicación incluida en la documentación de este manual, en la siguiente sección se dan detalles para el uso de esta aplicación, y en A.8.3 se describen los requisitos para hacer las configuraciones desde otra aplicación capaz de enviar datagramas UDP.

Un LED de color amarillo se enciende y apaga cada minuto, éste indica la interrupción interna del Ethmesh que refresca la tabla de direcciones. Este LED también es una ayuda visual para confirmar la correcta operación del nodo. La interfaz mini-USB es para conectar el hardware al Ethmesh y descargar el software.

Por otro lado la antena de caucho debe estar colocada para garantizar una adecuada comunicación con los demás nodos. También están disponibles unos pines en la parte inferior del Ethmesh, los dos últimos pines de ese conector tipo “pinhead” son para la alimentación de 3,3Vdc (ver A.9.2).

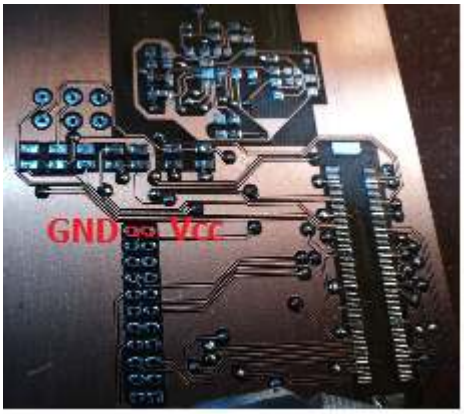


Figura A.8.2 Nodo Ethmesh.

Para conectar el nodo de manera directa con la computadora, sin un enrutador de por medio, debe asegurarse de estar en la misma red que el nodo Ethmesh. También debe prestar especial atención si está conectándose a una red con direcciones fijas, de ser necesario cambia la dirección IP de su computador o haga las configuraciones necesarias en el enrutador. Recuerde que puede probar la conectividad con el nodo realizando un PING a la dirección 192.168.1.50 desde la consola de comandos de Windows.

A.8.2 *Uso del Ethmesh usando la aplicación PCSocket*

En "Microchip Solutions" se encuentra la aplicación PCSocket, para correrla sólo debe hacer doble click sobre el ícono. Aparecerá la pantalla siguiente:

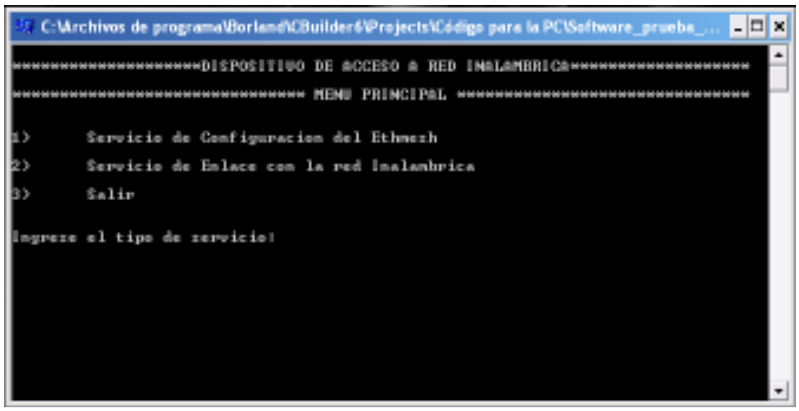


Figura A.8.3 Menú principal.

La opción 1) ingresa al menú de configuraciones y la opción 2) inicia el servicio de enlace con la red inalámbrica, 3) se utiliza para cerrar la aplicación. Si se ingresa al menú de configuraciones se le presentan las siguientes opciones:

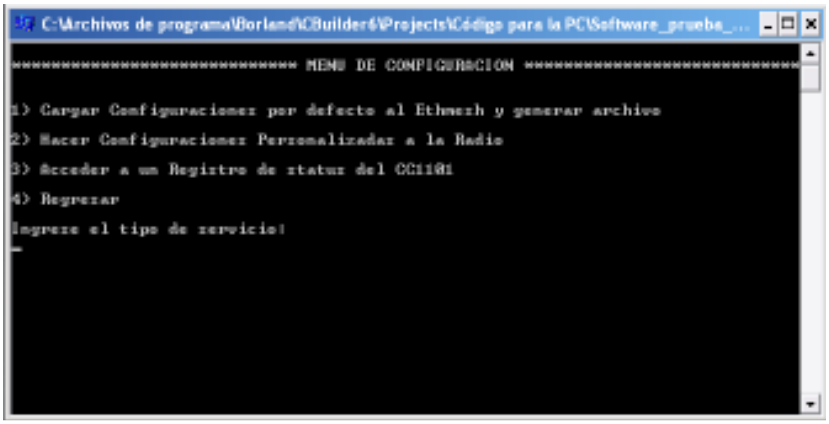


Figura A.8.4 Menú de configuraciones.

La opción 1) realiza el envío de un datagrama que contiene un mensaje con el texto "Acumine", esta palabra clave funciona para que el Ethmesh realice la operación de configuración inicial de la etapa de radiofrecuencia, también genera un archivo de texto con los valores que lee desde los registros del CC1101, esto le permite el usuario verificar la configuración que ha cargado al transductor.

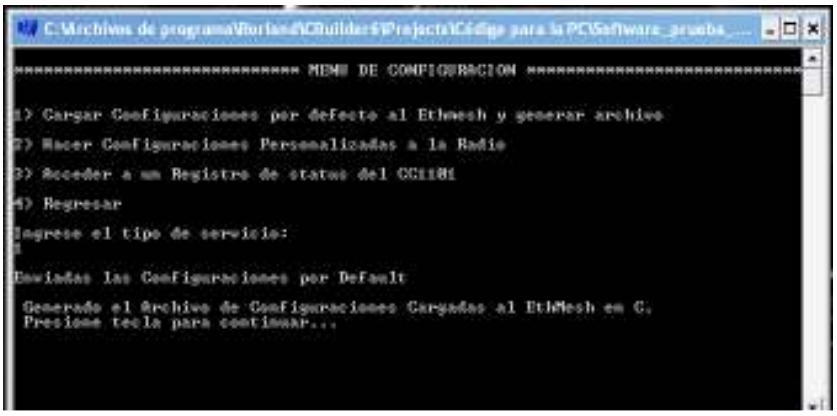


Figura A.8.5 Notificación de archivo generado.

Como puede verse en la figura anterior la aplicación notifica que se ha generado un archivo en la raíz, dicho archivo se crea por defecto con el nombre "Configs.txt". Usted puede abrirlo y revisar los 49 valores cargados en los registros, la manera en que se despliegan es de dos en dos y en orden consecutivo (ver figura A.8.6).

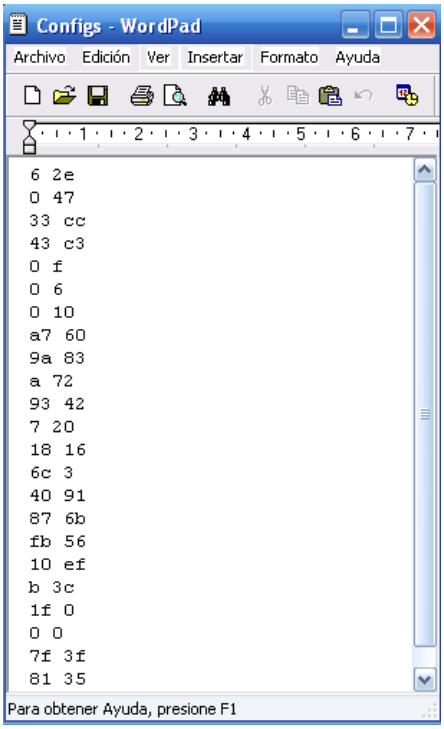


Figura A.8.6 Archivo con los valores para cada registro del CC1101.

Posteriormente usted podría realizar el ingreso al menú de configuraciones personalizadas, para ello regrese al menú oprimiendo *cualquier_tecla* + *enter*. Luego seleccione 2) e ingrese al menú de configuraciones personalizadas, que se muestra en la figura A.8.7.

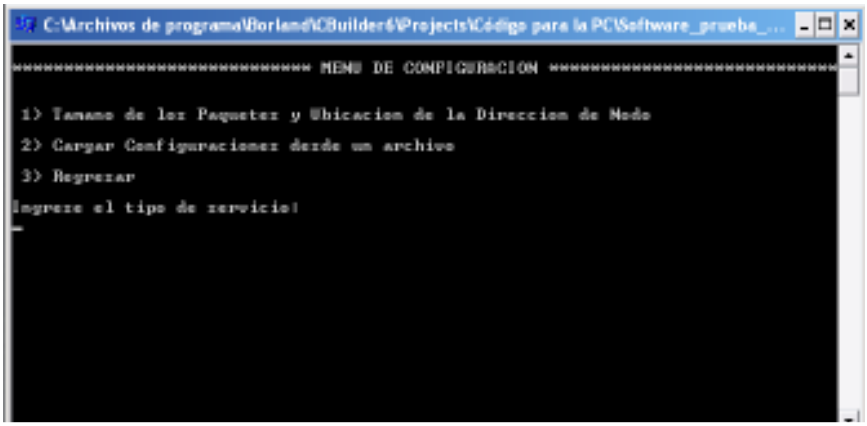


Figura A.8.7 Menú de configuraciones personalizadas.

En el menú de configuraciones personalizadas usted puede re-definir el tamaño del paquete y la ubicación del campo para la dirección destino. Ingrese 1) para configurar el tamaño (recuerde que el tamaño por defecto para las comunicaciones Acumine es de 67 bytes), o ingrese 2) para cargar las configuraciones desde un archivo de texto. Si usted escoge la primera opción se le presenta un menú como el de la siguiente figura.

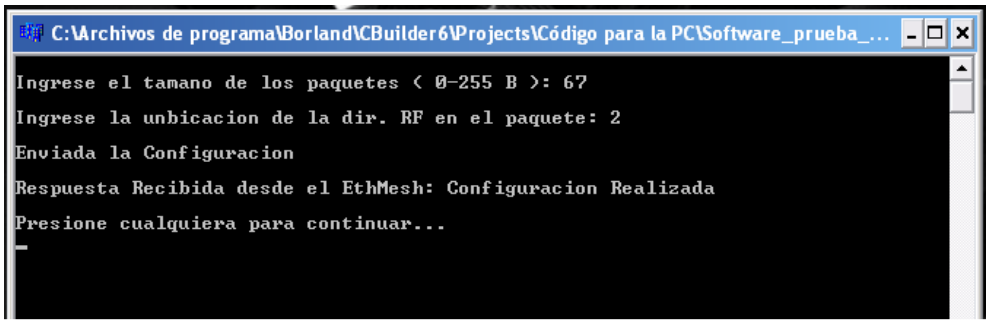


Figura A.8.8 Respuesta desde el Ethmesh luego de la configuración de paquete.

Como puede verse luego de ingresar un tamaño para el paquete (en esta imagen se uso el mismo) y una ubicación para la dirección destino, aparece un mensaje en la parte inferior indicando la respuesta recibida desde el nodo de enlace. Esta confirma le puede funcionar como verificación, en caso de error para cargar esta configuración el nodo le notificará con un mensaje.

Para cargar las configuraciones desde un archivo regrese al menú anterior, el que se muestra en la figura A.8.7 y seleccione la segunda opción. En la raíz usted tendrá a disposición un archivo llamado *archivo_config.txt*, éste puede ser usado como plantilla. Por favor no borre o incorpore nuevos nombres de registros, este documento es una ayuda visual para la configuración, se recomienda dirigirse a la hoja de datos del CC1101 antes de modificar los valores de los registros. Los valores en de los registros están en representación hexadecimal y son de tamaño un byte, los registros se han agrupado según su función. Una pequeña descripción aparece arriba de cada subgrupo, cambie el número

hexadecimal luego de estar seguro de las implicaciones del cambio en el comportamiento del transductor.

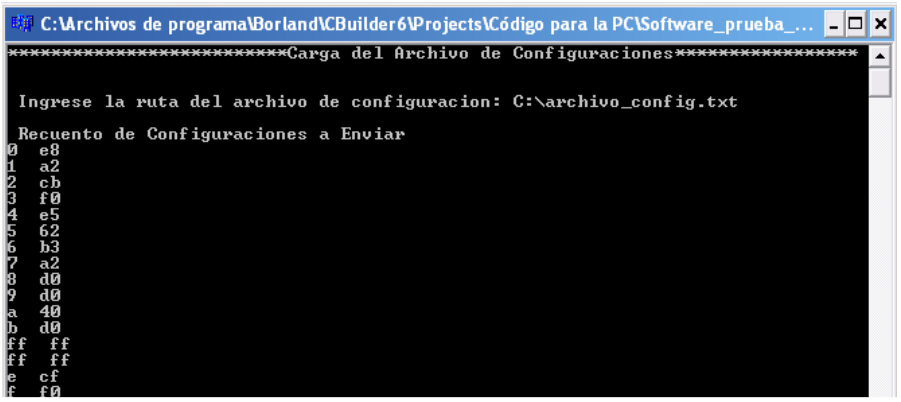


Figura A.8.9 Respuesta desde el Ethmesh luego de la configuración de paquete.

Tal como muestra la figura A.8.9, debe ingresar la ruta del archivo de texto que usará como plantilla, utilice sólo este archivo de texto para dicho propósito, ya que si usa cualquier otro archivo realizado por usted la aplicación PCSocket leerá de manera incorrecta los valores y cargará valores incoherentes en los registros del CC1101. Nótese que la aplicación le despliega una lista con los valores que usted modificó en el archivo de texto, esta última confirmación es para evitar que se mande por error una configuración inadecuada.

Ahora que se ha configurado el nodo, regrese al menú de configuraciones principal (ver A.8.4), este menú también ofrece la opción de acceder a un registro de estatus del CC1101, esta función es de mucha utilidad para depurar errores en la máquina de estados del transductor que pueden generarse tras realizar las configuraciones anteriores. En la figura A.8.10 aparecen las cuatro opciones para la manipulación remota del CC1101.

Figura A.8.10 Acceso remoto a los registros de estatus del CC1101.

En la primera opción aparece el registro de estado de la máquina de control interna del transductor, esta opción le permite verificar si el CC1101 se encuentra en estado de transmisión, recepción, espera, desborde de la RX_FIFO o underflow de la FIFO_TX. Como se observa en la siguiente figura, el Ethmesh contesta con el valor decimal correspondiente al estado del transductor.

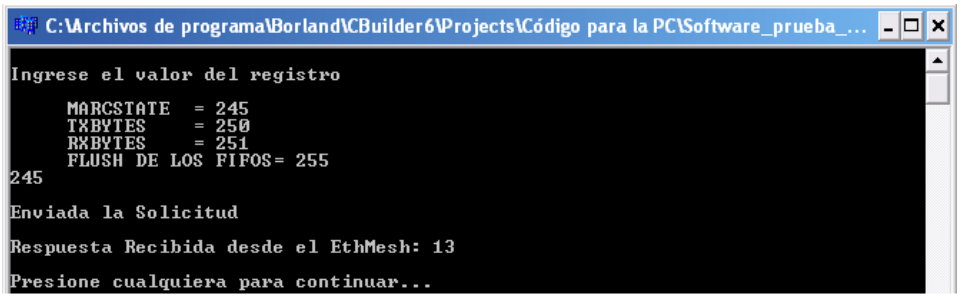


Figura A.8.11 Respuesta desde el Ethmesh a una petición del registro de estado.

En este ejemplo, el Ethmesh ha contestado con el valor decimal 13, haga uso de la tabla en la página 91 de la hoja de datos del CC1101, así podrá determinar el estado que corresponde al valor registrado por el nodo. Tenga en cuenta que en caso de detectar un estado de desborde (valor 17 decimal), puede realizar una purga de las memorias del transductor con la opción 255.

Ahora que ha revisado todas las opciones de configuración, retorne al menú principal. Seleccione la segunda opción e ingrese al menú de accesos a la red RF.

Figura A.8.12 Menú de accesos a la red RF.

La opción 1) le permite realizar transmisiones hacia un camión identificado con una dirección específica, o hacer una multidifusión a todos los camiones de la red. Note que el menú le indica cual dirección debe ser usada para multidifusiones. Por otro lado, un detalle muy importante en el funcionamiento de este software es que, el contenido de los paquetes debe ser cargado desde un archivo, este archivo puede ser de texto o binario.

En el ejemplo de la figura A.8.13 se hace el envío de un archivo binario de 568 bytes, para una configuración previa de paquetes de 63 bytes de tamaño, el software debe responsabilizarse de partir el contenido del archivo en fracciones no mayores al tamaño límite de un paquete RF. Además la aplicación es responsable de agregar el encabezado y el CRC para cada fracción del archivo.

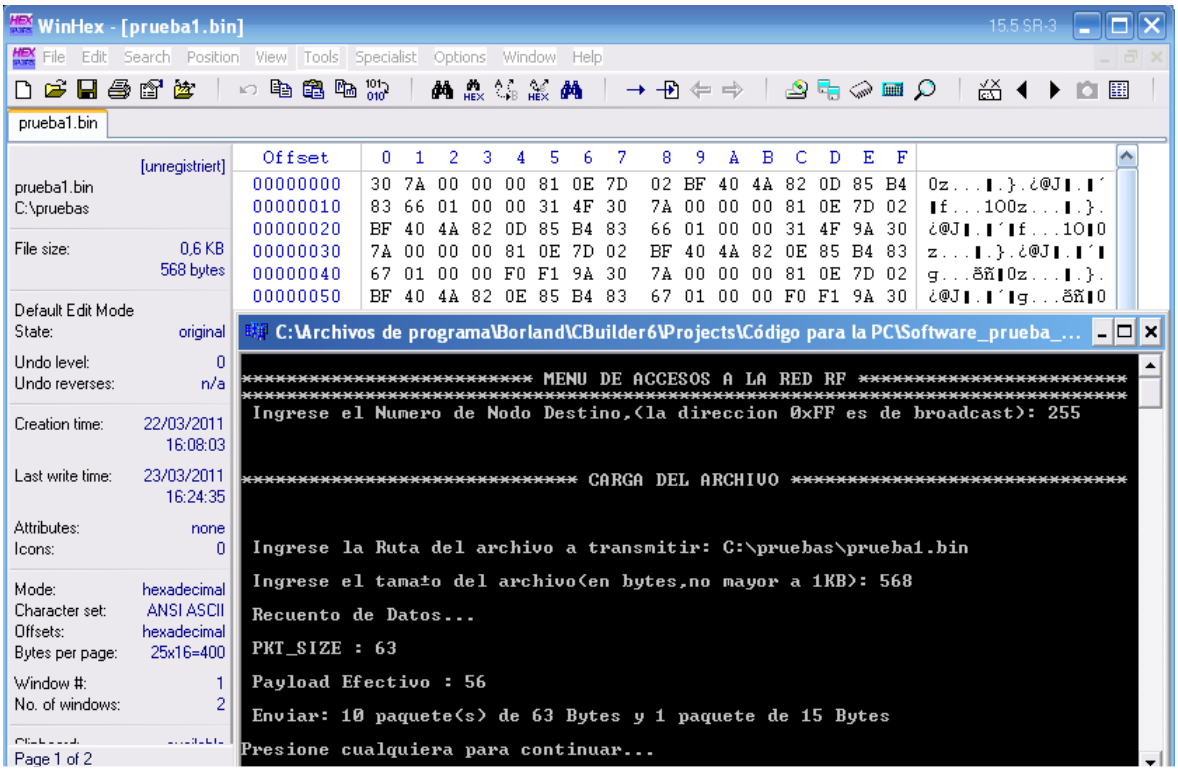


Figura A.8.13 Menú de envío de paquetes desde un archivo.

En el ejemplo de la figura se encuentra abierto el archivo binario con el editor de archivos binario WinHex, observe que a la izquierda de la figura se tiene que el tamaño de la figura es de 568 bytes, este valor corresponde al valor que usted como usuario del software debe especificar. Con ese valor y el valor de la carga efectiva, el software realiza los cálculos de cuántos paquetes debe enviar, también agrega el encabezado con formato Acumine. Recuerde que en éste va colocado primero la dirección, el tipo, el grupo y el tamaño de la carga efectiva.

De los 63 bytes que en este ejemplo se han predefinido, deben restarse 5bytes de encabezado y 2bytes de CRC. Por ende, queda un espacio de 56bytes para acomodar el contenido del archivo binario. Entonces 10 paquetes con payload de 56bytes y uno de 8bytes completan el total de 568 bytes que mide el archivo que desea enviarse en este ejemplo. Una vez que se presenta un recuento de datos a enviar, el software procede con el envío y reporta su culminación (ver figura A.8.14).

Como puede verse, existe la posibilidad de continuar enviando transmisiones a algún vehículo o regresar al menú principal. Ingrese “n” si desea regresar o “y” si desea continuar enviando.

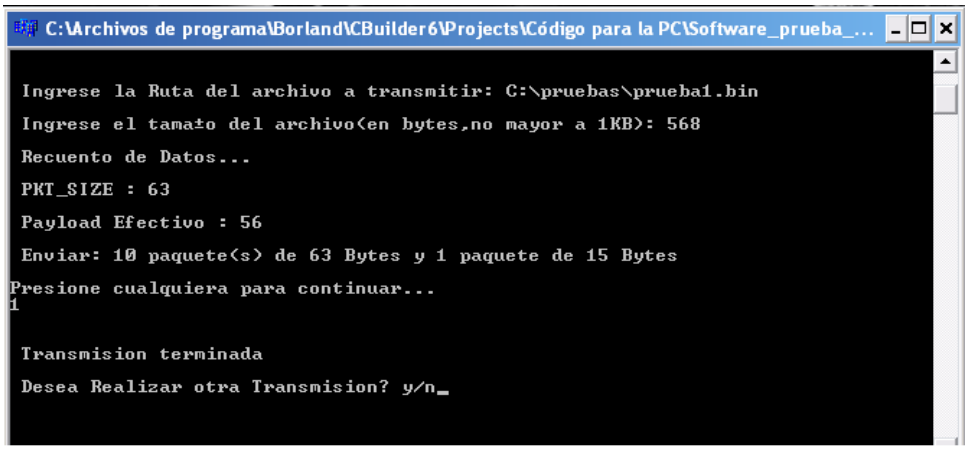


Figura A.8.14 Final de transmisión desde el menú.

Ahora analizaremos las opciones para la recepción de paquetes con el software. Regrese al menú de enlace (figura A.8.12) y seleccione la segunda opción. Lo primero que ofrece el menú es realizar un archivo de respaldo con los paquetes recibidos, de ser así debe ingresar la ruta para crear el archivo binario, una copia llamada "Copia.txt" se genera automáticamente para visualizar los valores de los bytes recibidos como texto.

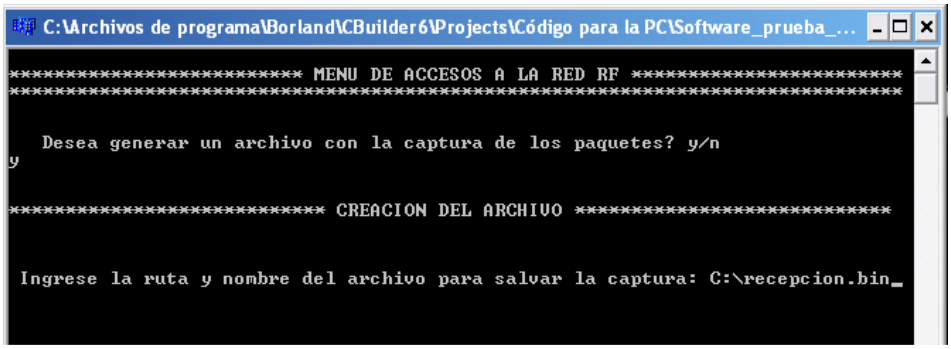


Figura A.8.15 Creación de un archivo para respaldo de la información recibida desde un camión.

En el ejemplo de la figura A.8.15 se eligió un archivo binario en la raíz, recuerde que se generará una copia de éste en un .txt. Posteriormente es necesario definir el tiempo de vida para el enlace de recepción, y una dirección para identificarse en la red RF.

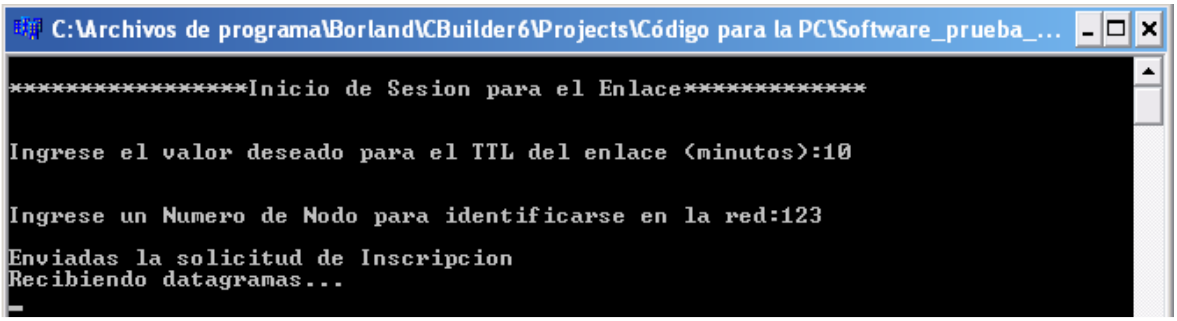


Figura A.8.16 Creación de un archivo para respaldo de la información recibida desde un camión.

Luego se empiezan a recibir los paquetes desde el Ethmesh que correspondan a la dirección que se ha asignado o bien, los paquetes de multidifusión transmitidos por algún camión. Luego de concluido el tiempo de duración del enlace, el Ethmesh manda un reporte al respecto y la aplicación cierra el archivo con la información y notifica al usuario.

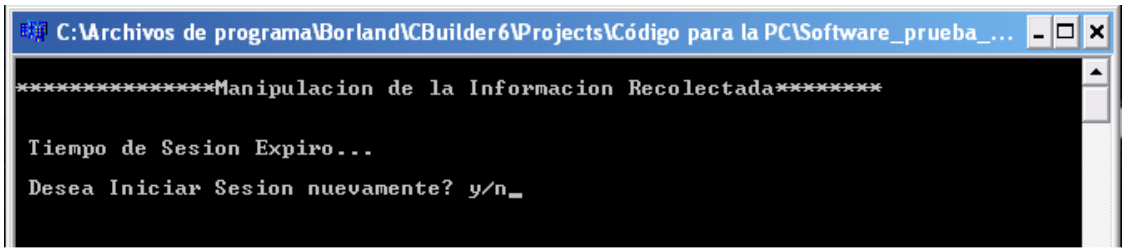


Figura A.8.17 Notificación sobre fin de sesión desde el Ethmesh.

En la figura A.8.18 se muestra un archivo binario recibido con varios paquetes, nótese que el encabezado corresponde al usado en Acumine, en celeste se ha encerrado el primer paquete recibido.

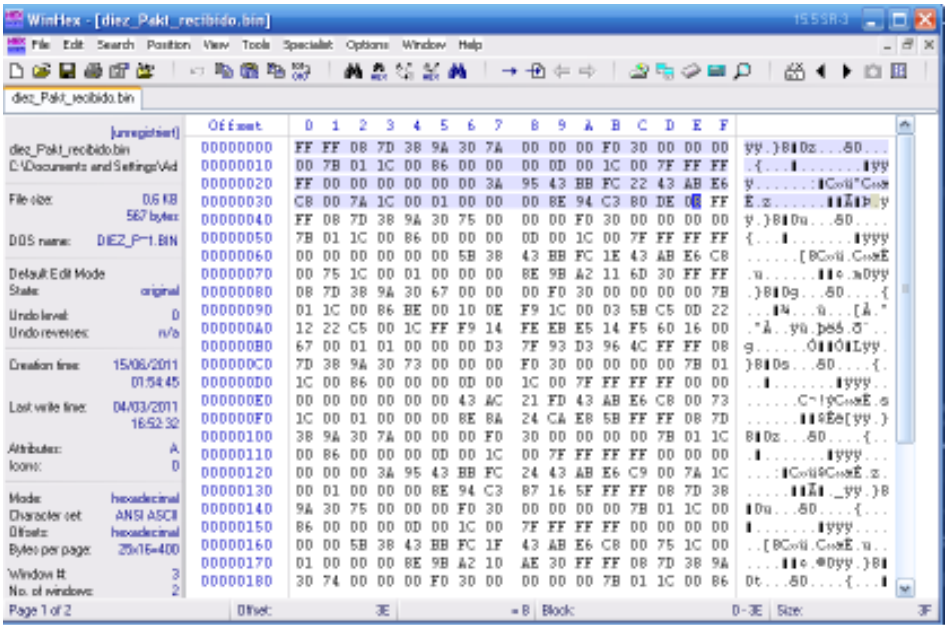


Figura A.8.18 Notificación sobre fin de sesión desde el Ethmesh.

Así se terminan de describir las funciones del software de prueba, en la siguiente sección se describen los procedimientos para la comunicación a través del Ethmesh utilizando otro software alternativo.

A.8.3 *Requisitos para el uso del Ethmesh con software alternativo*

La siguiente tabla resume el protocolo y puertos necesarios para la comunicación y configuración a través del Ethmesh, la idea es que sirva de guía para cualquiera que desee programar su propio software para manejar el nodo. El principal requisito es que sea capaz de enviar y recibir mensajes UDP y utilice los puertos 6651, 6652 y 6653.

Servicio	Descripción del servicio	Descripción del paquete y procedimiento	Respuesta esperada desde el Ethmesh	Puerto usado
Cargar configuraciones por defecto al CC1101.	Este servicio hace que el PIC cargue 47 valores a los registros de configuración, esta programación por defecto es compatible con las cajas Acumine.	Cargar los caracteres 'Acu' entre los bytes 0-2 del buffer de salida.	Los 47 valores hexadecimales de configuración son regresados en un datagrama.	6651
Cargar todas las configuraciones al CC1101, pero con valores personalizados.	Esta opción puede ser usada para enviar valores distintos a los compatibles con Acumine para cada registro de configuración. Es necesario que se carguen todos los valores.	Cargar los caracteres 'CONFI' entre los bytes 0-4 del buffer de salida. En los siguientes 94 bytes del datagrama, debe cargar desde el registro 0x00 hasta el 0x2E. Luego adjuntar el valor para el registro 3E.	"Reconfiguración del Nodo finalizada...". En caso de que la operación se cumpliera con éxito. "Reconfiguración del Nodo fallida...". En caso contrario.	6651
Cambiar el tamaño y ubicación de la dirección de nodo.	Conforme al tamaño de paquete programado, el Ethmesh recibirá o enviará la cantidad de veces necesarias para completar un paquete. La posición del nodo es configurable para poder trabajar con otros protocolos de paquete.	Escribir en las dos primeras posiciones 'PS', en la tercera posición cargue el tamaño en bytes para el paquete RF, con máximo 255. En el cuarto lugar escriba un valor para la posición de la dirección de nodo destino.	Un mensaje de "Configuración Realizada" si todo sale bien, en caso contrario espérese un mensaje "Configuración Tamaño Fallida...".	6651
Obtener el valor de un registro de estatus.	Esta opción puedes ser usada para inspeccionar el estado en que se encuentre el cc1101.	Se debe colocar 'RA' en las posiciones 0 y 1 del buffer de salida. Luego la dirección del registro de estatus que se desea leer.	Se regresará el valor hexadecimal del registro solicitado en la primera posición del datagrama.	6651

Servicio	Descripción del servicio	Descripción del paquete y procedimiento	Respuesta esperada desde el Ethmesh	Puerto usado
Hacer purga de las memorias FIFO del CC1101.	Se envía una instrucción para el vacío de la memoria y así sacar al transductor de estados de desborde, se limpiarán ambas FIFO.	Se debe cargar un string 'FR' en las primeras dos posiciones.	Un datagrama con un mensaje: "Flush Realizado con Exito" o "No se Realizaron las Configuraciones"	6651
Realizar una transmisión hacia la red RF.	El Ethmesh esperará únicamente un paquete con la información y encabezado listo. Enviará el paquete tal y como lo reciba, el usuario debe asegurarse de que el tamaño de éste coincida con el tamaño pre-configurado.	El procedimiento de solicitudes previas es opcional. Puede hacerse o enviar los paquetes en un datagrama de forma directa y sin consultar antes sobre el estado del buffer interno del Ethmesh.	Para realizar el procedimiento con solicitudes envíe un mensaje 'CANYOU' en un datagrama. Espere una respuesta, si el mensaje de vuelta en 'YES', entonces puede enviar el paquete para su transmisión, pero si la respuesta en 'NO', entonces es incierto si el paquete es enviado o no.	6653
Comenzar con una recepción desde la red RF.	El Ethmesh capturará todos los paquetes que sean para el número de nodo indicado por el usuario o los que sean de multidifusión, luego los hará llegar al computador correspondiente a esa dirección de nodo. Antes es necesaria una solicitud de inscripción a la tabla de enrutamiento que el dispositivo usa para regresar de vuelta los mensajes. Además debe enviar el valor en minutos del tiempo para la captura.	Envíe un mensaje 'START' entre las posiciones 0-4, luego concatene en 5 el valor de nodo que desea para el Ethmesh, y en 6 la cantidad de minutos para el enlace, ambos valores con un límite de 255.	Puede esperar que un mensaje 'LISTED' de vuelta si es inscrito en la tabla, además comenzará a recibir inmediatamente todos los paquetes correspondientes a su número de nodo o broadcast. En caso de no poder ser inscrito porque la tabla esté llena (tamaño máximo de la tabla 100 direcciones) entonces se recibirá un mensaje 'NOTLISTED'.	6651 para el envío de las solicitudes de inscripción desde la PC hasta el Ethmesh. 6653 únicamente usado por el Ethmesh para enviar los datagramas capturados de la red RF hacia la PC correspondiente.

Apéndice A.9 Glosario de abreviaturas.

10Base-T: Versión de Ethernet capaz de emplear velocidades de hasta 10Mbps.

100Base-T: Versión de Ethernet capaz de emplear velocidades de hasta 100Mbps.

2FSK: Modulación FSK binaria, es la denominación que se le da este tipo de modulación cuando se utilizan únicamente dos símbolos para la representación de los bits.

Acumine: Nombre bajo el cual se han registrado las tecnologías desarrolladas por el IIIE, para aplicaciones en minería a cielo abierto.

ADC: Convertidor analógico a digital, tiene como función la digitalización de una señal continua analógica. Permite la manipulación digital de señales de comunicación.

Ad Hoc: Topología de red inalámbrica dinámica, que permite rutas con forma variante, no depende de puntos de acceso para manejar las comunicaciones por la red.

Agilent E4433B: Analizador de espectros, utilizado para realizar mediciones de señales en alta frecuencia.

AODV: Esquema de enrutamiento bajo demanda, la formación de las rutas se basa en la petición de conexión por multidifusión por parte de los nodos que desean iniciar una comunicación.

API: Concepto de programación que se refiere a una aplicación que funciona como interfaz para manejar algún conjunto de funciones.

ARP: Protocolo para la resolución de direcciones usado cuando un dispositivo intermedio para conocer la dirección física de un dispositivo a partir de una dirección lógica.

Balun: Dispositivo usado para conexión de antenas no balanceadas a una entrada RF balanceada.

Broadcast: Transmisión dirigida a todos los miembros de una red, también llamada multidifusión.

CANBus: Interfaz de conexión para unidades electrónicas, muy usado en aplicaciones industriales electromecánicas, presenta gran inmunidad a las interferencias propias de ambientes agresivos.

CC1101: Modelo de transductor de radiofrecuencia de la marca Texas Instruments y utilizado por las tecnologías Acumine.

CRC: Algoritmo de verificación del contenido de un paquete por redundancia cíclica.

CSMA/CD: Método para compartir un mismo canal con varios dispositivos, consiste en la detección de un nivel de energía en el canal como señal de que existe una comunicación en curso, el método espera por un tiempo aleatorio luego de una colisión para intentar transmitir de nuevo.

DIEC: Departamento de Ingeniería Eléctrica y de Computadores de la UNS.

Ethmesh: Nombre con que se le ha dado al nodo de enlace diseñado en este proyecto.

Ethernet: Estándar que define las dos primeras capas del modelo TCP/IP. Define la parte lógica del manejo del canal y ha sido modificada en varias ocasiones para adaptarse a varios medios físicos de conexión.

FIFO: Memoria en la que el primer dato en ser almacenado es el primero en ser leído.

Footprint: Se refiere al diseño para el montaje de un componente en un PCB.

FSK: Modulación digital por frecuencia, representa una señal binaria en símbolos que pueden o no estar codificados.

GPS: Sistema de posicionamiento global por satélite, usa dispositivos electrónicos y satélites para la ubicación de objetos en la superficie terrestre con una precisión de hasta centímetros.

Headers: Encabezados, son un tipo de archivo en lenguaje de programación C, contienen definiciones importantes para el resto del código fuente.

IIIE: Instituto de Investigaciones en Ingeniería Eléctrica, fundado por Alfredo Desages con profesores y becarios del DIEC.

I/Q: Componentes que se obtienen como producto de la descomposición de una señal en un conjunto de dos señales que son independientes, una de ellas es conocida como la componente I, que quiere decir en fase; y la componente Q, que quiere decir en cuadratura.

ISM: Estándar de frecuencias de comunicación para uso médico, industrial y de propósitos científicos.

IP: Protocolo de internet, utilizado en comunicaciones sobre medios físicos Ethernet, inalámbricos y actualmente de fibra óptica. Es un protocolo de capa de red del modelo TCP/IP.

J-TAG: Interfaz de programación y depurado para microcontroladores.

MAC: Control de acceso al medio, se refiere a los procedimientos y protocolos que intervienen en el uso de un determinado canal de comunicación.

Manchester: Codificación utilizada en comunicaciones digitales para nivelación del nivel CD en una señal binaria.

Mesh: es una clase de topología de red inalámbrica, mezcla entre una topología de infraestructura y una dinámica.

MIPS32: Arquitectura de procesadores embebidos con un set reducido pero robusto de instrucciones.

MII: Interfaz de acceso al medio, independiente del medio. Utilizada para comunicar el controlador de red al transductor LAN.

MPR: Nodo seleccionado por un vecino para encargarse de la selección y mantenimiento de las rutas de comunicación.

MSO7104A: Osciloscopio de señales mixtas, puede usarse para señales analógicas y digitales en frecuencias de hasta 1GHz.

Multicast: Conocido también como multidifusión, consiste en mensajes que son emitidos por un nodo pero que van dirigidos todos los integrantes de una red.

LAN: Red de área local.

OLSR: Protocolo para redes móviles que define rutas de manera previa, su funcionamiento se basa en un grupo selecto de nodos vecinos para la formación de una ruta.

OSI: Modelo de referencia para la teoría de redes.

Overflow: Relacionado al estado de una memoria, se refiere al desbordamiento por un exceso en la cantidad de información almacenada en una memoria.

PCB: Circuito impreso en una tableta con una superficie conductora de cobre que se usa para trazar las líneas de conexión entre los componentes.

PIC32MXX75F256L: Microcontrolador de la marca Microchip, tiene una arquitectura MIPS de 32bits, capacidades de manejo para distintas interfaces de comunicación.

PKS: Modulación digital por fase, modifica la fase de una señal analógica para imprimir la información en una portadora.

Polling: Método para inspeccionar el estado de un evento, el cual consiste en una revisión constante del mismo.

PQI: Indicador usado por un protocolo de comunicación para evaluar la calidad de un preámbulo recibido previo a una recepción.

PQT: Relación entre un indicador para la calidad del preámbulo y un valor pre-establecido.

RCF: Archivos públicos que se encuentran disponibles en la red para ser usados como referencia, contienen información sobre las actualizaciones realizadas a varios protocolos y estándares de comunicación.

RF: Radiofrecuencia, se refiere a frecuencias muy elevadas utilizadas en las comunicaciones inalámbricas.

RJ-45: Tipo de conector que vino a sustituir a los conectores coaxiales que presentaban algunas deficiencias físicas. Utilizado en redes Ethernet basadas en cable trenzado.

RMII: Interfaz independiente del medio físico, versión reducida con menor cantidad de señales que MII.

RS-232: Interfaz serial de comunicación, puede ser síncrona o asíncrona.

Socket BSD: Interfaz de programación que comprende una librería para el desarrollo de aplicaciones en lenguaje de programación C.

SPI: Interfaz de comunicación serial que comparte el mismo bus de datos con varios dispositivos.

SRDs: Grupo de dispositivos que utilizan una comunicación inalámbrica de corta distancia.

Stack: En el contexto de este trabajo, se refiere a un conjunto de librerías y código fuente que facilita la invocación de API´s.

Strobe: En el contexto de este documento, se refiere a una instrucción o comando enviado a través de una interfaz SPI.

TCP/IP: Modelo de protocolos que define las funciones de cada etapa en una comunicación por red.

TCP: Protocolo de control de transferencias, pertenece a la capa de transporte del modelo TCPI/IP.

Threshold: Nivel de umbral que determina la actualización de un estado o evento.

UDP: Protocolo de la capa de transporte del modelo TCP/IP. Realiza transmisiones de datagramas.

Unicast: Transmisión dirigida de un punto a otro, pueden intervenir varios dispositivos en el enrutamiento del mensaje, pero el mensaje es únicamente para un destinatario.

UNS: Universidad Nacional del Sur, universidad pública de la república de Argentina, ubicada en la ciudad de Bahía Blanca, provincia de Buenos Aires. Catalogada como de las mejores universidades públicas del país.

UTP: Clase de cable utilizado para las comunicaciones Ethernet, cuenta con un trenzado entre pares para evitar la interferencia electromagnética generada en alta frecuencia.

WRT54G: Modelo de enrutador de la marca Linksys, uno de los más utilizados y cómodos hasta el momento.

Anexos

Anexo B.1 Enrutador WRT54G de Linksys.

Anexo B.2 Características generales del transductor CC1000. Tomado de [40].

Chipcon SmartRF® CC1000

CC1000
Single Chip Very Low Power RF Transceiver

Applications

- *Very low power UHF wireless data transmitters and receivers*
- *315 / 433 / 868 and 915 MHz ISM/SRD band systems*
- *RKE – Two-way Remote Keyless Entry*
- *Home automation*
- *Wireless alarm and security systems*
- *AMR – Automatic Meter Reading*
- *Low power telemetry*
- *Toys*

Product Description

CC1000 is a true single-chip UHF transceiver designed for very low power and very low voltage wireless applications. The circuit is mainly intended for the ISM (Industrial, Scientific and Medical) and SRD (Short Range Device) frequency bands at 315, 433, 868 and 915 MHz, but can easily be programmed for operation at other frequencies in the 300-1000 MHz range.

The main operating parameters of **CC1000** can be programmed via an easy-to-interface serial bus, thus making **CC1000** a very flexible and easy to use transceiver. In a typical system **CC1000** will be used together with a microcontroller and a few external passive components.

CC1000 is based on Chipcon's SmartRF® technology in 0.35 µm CMOS.

Features

- True single chip UHF RF transceiver
- Very low current consumption
- Frequency range 300 – 1000 MHz
- Integrated bit synchroniser
- High sensitivity (typical -110 dBm at 2.4 kBaud)
- Programmable output power –20 to 10 dBm
- Small size (TSSOP-28 package)
- Low supply voltage (2.1 V to 3.6 V)
- Very few external components required
- No external RF switch / IF filter required
- RSSI output
- Single port antenna connection
- FSK data rate up to 76.8 kBaud
- Complies with EN 300 220 and FCC CFR47 part 15
- FSK modulation spectrum shaping
- Programmable frequency in 250 Hz steps makes crystal temperature drift compensation possible without TCXO
- Suitable for frequency hopping protocols
- Development kit available
- Easy-to-use software for generating the **CC1000** configuration data

Anexo B.3 Características generales del transductor CC1101. Tomado de [4].

Chipcon Products
from Texas Instruments

CC1101

CC1101
Low-Cost Low-Power Sub-1GHz RF Transceiver (Enhanced CC1100)

Applications

- *Ultra low-power wireless applications operating in the 315/433/868/915 MHz ISM/SRD bands*
- *Wireless alarm and security systems*
- *Industrial monitoring and control*
- *Wireless sensor networks*
- *AMR – Automatic Meter Reading*
- *Home and building automation*

Product Description

The CC1101 is a low-cost sub- 1 GHz transceiver designed for very low-power wireless applications. The circuit is mainly intended for the ISM (Industrial, Scientific and Medical) and SRD (Short Range Device) frequency bands at 315, 433, 868, and 915 MHz, but can easily be programmed for operation at other frequencies in the 300-348 MHz, 387-464 MHz and 779-928 MHz bands.

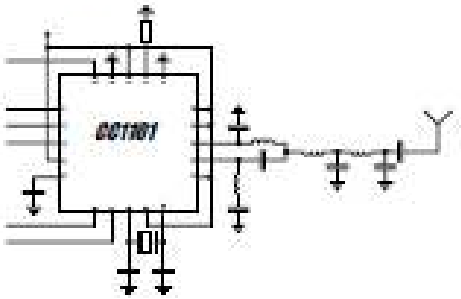
CC1101 is an improved and code compatible version of the CC1100 RF transceiver. The main improvements on the CC1101 include:

- Improved spurious response
- Better close-in phase noise improving Adjacent Channel Power (ACP) performance
- Higher input saturation level
- Improved output power ramping
- Extended frequency bands of operation, i.e.
  CC1100: 400-464 MHz and 800-928 MHz
  CC1101: 387-464 MHz and 779-928 MHz

The RF transceiver is integrated with a highly configurable baseband modem. The modem supports various modulation formats and has a configurable data rate up to 500 kBaud.

CC1101 provides extensive hardware support for packet handling, data buffering, burst transmissions, clear channel assessment, link quality indication, and wake-on-radio.

The main operating parameters and the 64-byte transmit/receive FIFOs of CC1101 can be controlled via an SPI interface. In a typical system, the CC1101 will be used together with a microcontroller and a few additional passive components.



This product shall not be used in any of the following products or systems without prior express written permission from Texas Instruments:

(i) implantable cardiac rhythm management systems, including without limitation pacemakers, defibrillators and cardiac resynchronization devices,

(ii) external cardiac rhythm management systems that communicate directly with one or more implantable medical devices; or

(iii) other devices used to monitor or treat cardiac function, including without limitation pressure sensors, biochemical sensors and neurostimulators.

Please contact lpw-medical-approval@list.ti.com if your application might fall within the category described above.

Key Features

RF Performance

- High sensitivity (–111 dBm at 1.2 kBaud, 868 MHz, 1% packet error rate)
- Low current consumption (14.7 mA in RX, 1.2 kBaud, 868 MHz)
- Programmable output power up to +10 dBm for all supported frequencies
- Excellent receiver selectivity and blocking performance
- Programmable data rate from 1.2 to 500 kBaud
- Frequency bands: 300-348 MHz, 387-464 MHz and 779-928 MHz

Analog Features

- 2-FSK, GFSK, and MSK supported as well as OOK and flexible ASK shaping
- Suitable for frequency hopping systems due to a fast settling frequency synthesizer: 90us settling time
- Automatic Frequency Compensation (AFC) can be used to align the frequency synthesizer to the received center frequency
- Integrated analog temperature sensor

Digital Features

- Flexible support for packet oriented systems: On-chip support for sync word detection, address check, flexible packet length, and automatic CRC handling
- Efficient SPI interface: All registers can be programmed with one "burst" transfer
- Digital RSSI output
- Programmable channel filter bandwidth
- Programmable Carrier Sense (CS) indicator
- Programmable Preamble Quality Indicator (PQI) for improved protection against false sync word detection in random noise
- Support for automatic Clear Channel Assessment (CCA) before transmitting (for listen-before-talk systems)
- Support for per-package Link Quality Indication (LQI)
- Optional automatic whitening and de-whitening of data

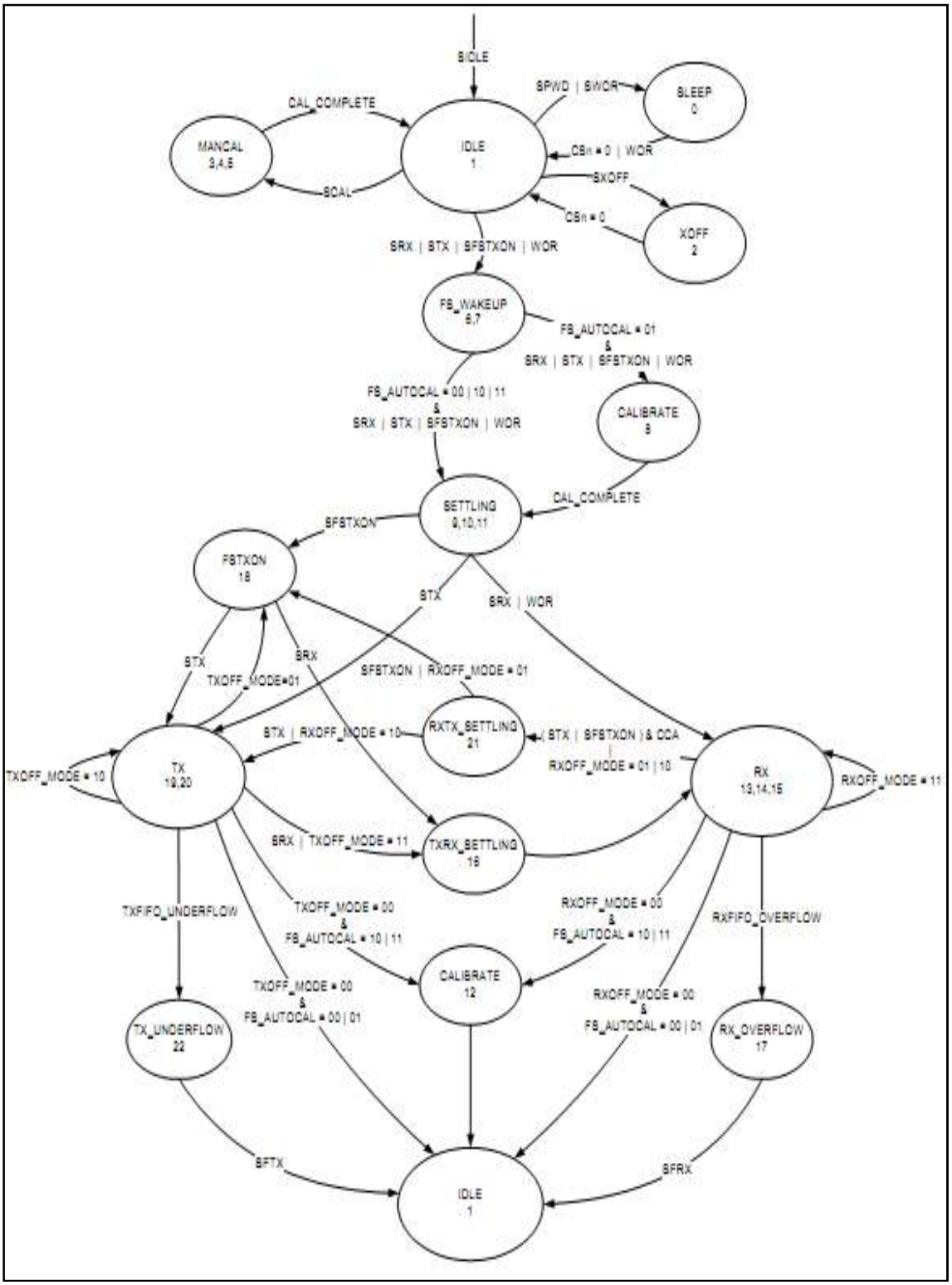
Low-Power Features

- 400 nA sleep mode current consumption
- Fast startup time: 240us from sleep to RX or TX mode (measured on EM reference design [5] and [6])
- Wake-on-radio functionality for automatic low-power RX polling
- Separate 64-byte RX and TX data FIFOs (enables burst mode data transmission)

General

- Few external components: Completely on-chip frequency synthesizer, no external filters or RF switch needed
- Green package: RoHS compliant and no antimony or bromine
- Small size (QLP 4x4 mm package, 20 pins)
- Suited for systems targeting compliance with EN 300 220 (Europe) and FCC CFR Part 15 (US).
- Support for asynchronous and synchronous serial receive/transmit mode for backwards compatibility with existing radio communication protocols

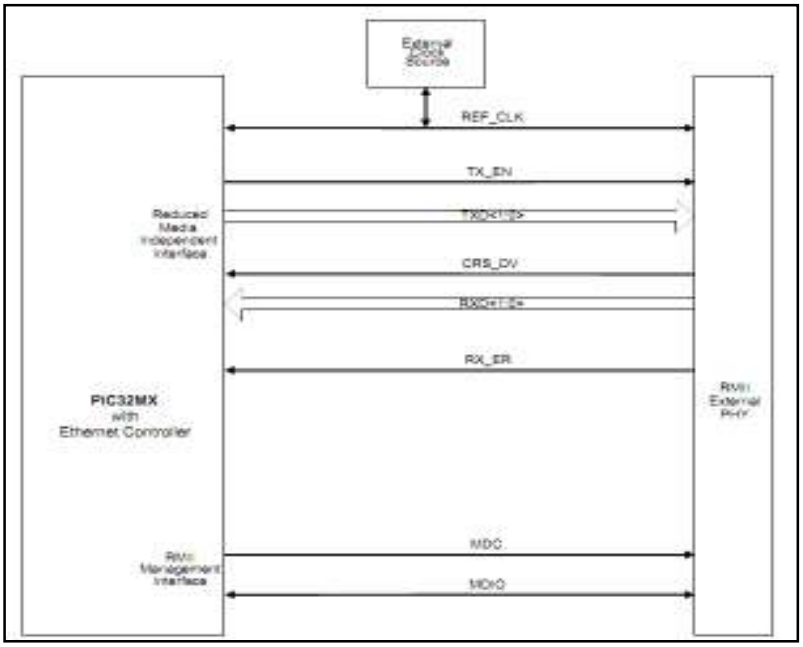
Anexo B.4 Diagrama completo de estados del CC1101. Tomado de [40].



Anexo B.5 Tabla de estados con su correspondiente valor asignado en el registro de estado. Tomado de [40].

Value	State name	State (Figure 22, page 48)
0 (0x00)	SLEEP	SLEEP
1 (0x01)	IDLE	IDLE
2 (0x02)	XOFF	XOFF
3 (0x03)	VCOON_MC	MANCAL
4 (0x04)	REGON_MC	MANCAL
5 (0x05)	MANCAL	MANCAL
6 (0x06)	VCOON	FS_WAKEUP
7 (0x07)	REGON	FS_WAKEUP
8 (0x08)	STARTCAL	CALIBRATE
9 (0x09)	BWBOOST	SETTLING
10 (0x0A)	FS_LOCK	SETTLING
11 (0x0B)	IFADCON	SETTLING
12 (0x0C)	ENDCAL	CALIBRATE
13 (0x0D)	RX	RX
14 (0x0E)	RX_END	RX
15 (0x0F)	RX_RST	RX
16 (0x10)	TXRX_SWITCH	TXRX_SETTLING
17 (0x11)	RXFIFO_OVERFLOW	RXFIFO_OVERFLOW
18 (0x12)	FSTXON	FSTXON
19 (0x13)	TX	TX
20 (0x14)	TX_END	TX
21 (0x15)	RXTX_SWITCH	RXTX_SETTLING
22 (0x16)	TXFIFO_UNDERFLOW	TXFIFO_UNDERFLOW

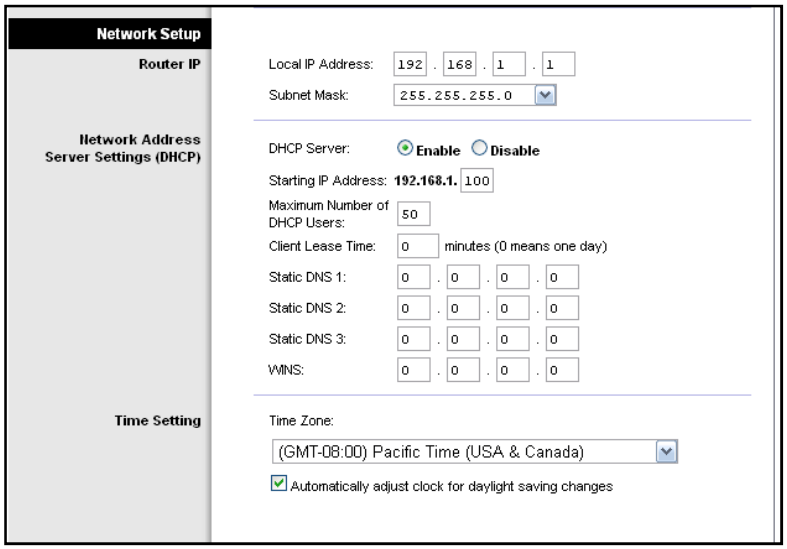
Anexo B.6 Interfaz de comunicación RMII. Tomado de [43]



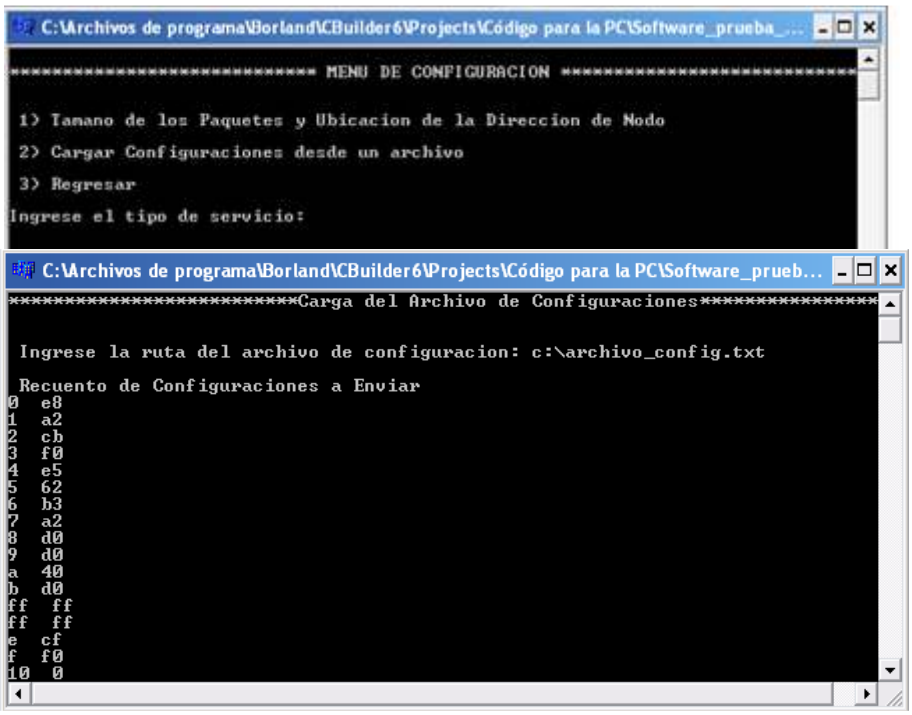
Anexo B.7 Mapa de los registros del CC1101. Tomado de [40]

	Write		Read		
	Single Byte	Burst	Single Byte	Burst	
	+0x00	+0x40	+0x80	+0xC0	
0x00	IOCFG2				R/W configuration registers, burst access possible
0x01	IOCFG1				
0x02	IOCFG0				
0x03	FIFOTHR				
0x04	SYNC1				
0x05	SYNC0				
0x06	PKTLEN				
0x07	PKTCTRL1				
0x08	PKTCTRL0				
0x09	ADDR				
0x0A	CHANNR				
0x0B	FSCTRL1				
0x0C	FSCTRL0				
0x0D	FREQ2				
0x0E	FREQ1				
0x0F	FREQ0				
0x10	MDMCFG4				
0x11	MDMCFG3				
0x12	MDMCFG2				
0x13	MDMCFG1				
0x14	MDMCFG0				
0x15	DEVIATN				
0x16	MCSM2				
0x17	MCSM1				
0x18	MCSM0				
0x19	FOCCFG				
0x1A	BSCFG				
0x1B	AGCCTRL2				
0x1C	AGCCTRL1				
0x1D	AGCCTRL0				
0x1E	WOREVT1				
0x1F	WOREVT0				
0x20	WORCTRL				
0x21	FREND1				
0x22	FREND0				
0x23	FSCAL3				
0x24	FSCAL2				
0x25	FSCAL1				
0x26	FSCAL0				
0x27	RCCTRL1				
0x28	RCCTRL0				
0x29	FSTEST				
0x2A	PTEST				
0x2B	AGCTEST				
0x2C	TEST2				
0x2D	TEST1				
0x2E	TEST0				
0x2F					
0x30	SRES		SRES	PARTNUM	Command Strobes, Status registers (read only) and multi byte registers
0x31	SFSTXON		SFSTXON	VERSION	
0x32	SXOFF		SXOFF	FREQEST	
0x33	SCAL		SCAL	LQI	
0x34	SRX		SRX	RSSI	
0x35	STX		STX	MARCSTATE	
0x36	SIDLE		SIDLE	WORTIME1	
0x37				WORTIME0	
0x38	SWOR		SWOR	PKTSTATUS	
0x39	SPWD		SPWD	VCO_VC_DAC	
0x3A	SFRX		SFRX	TXBYTES	
0x3B	SFTX		SFTX	RXBYTES	
0x3C	SWORRST		SWORRST	RCCTRL1_STATUS	
0x3D	SNOP		SNOP	RCCTRL0_STATUS	
0x3E	PATABLE	PATABLE	PATABLE	PATABLE	
0x3F	TX FIFO	TX FIFO	RX FIFO	RX FIFO	

Anexo B.8 Configuración del enrutador.



Anexo B.9 Menú y Submenú para configuraciones personalizadas

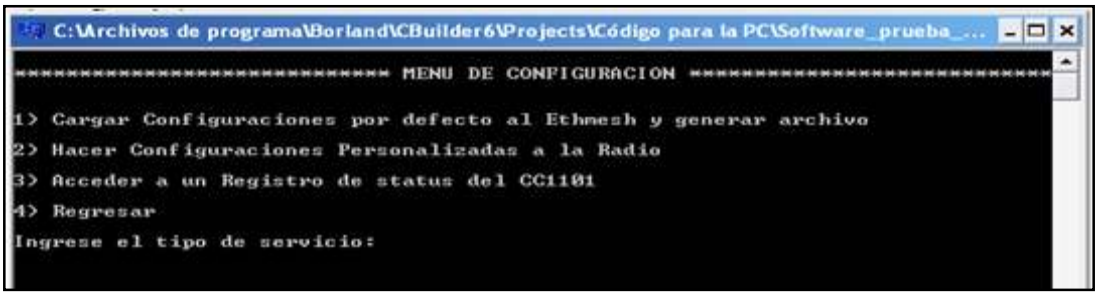


Anexo B.10 Plantilla para la configuración por registros del Ethmesh.

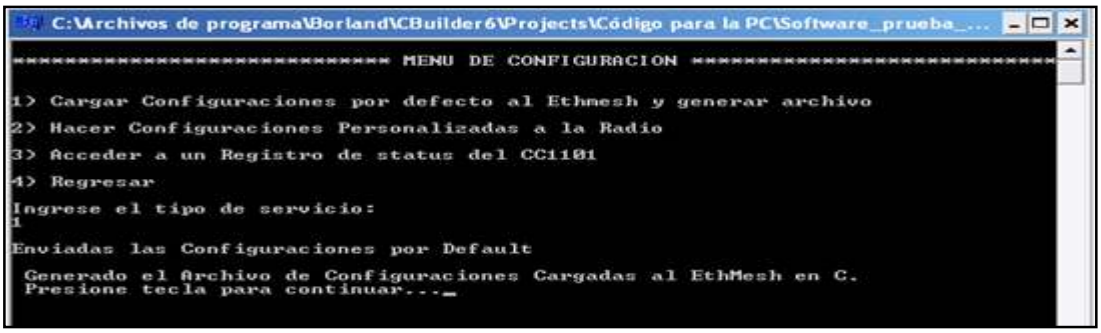
```
**********Archivo de Configuraciones del EthMesh********
IMPORTANTE: Por Favor no borre o incorpore nuevos nombres de registros,
            este documento es una ayuda visual para la configuración.
            Se recomienda dirigirse a la hoja de datos del CC1101 antes de
            modificar los valores de los registros. Los siguientes valores
            están en representación hexadecimal y son de tamaño un byte.

//Pines de status límite de los buffers FIFO
IOCFG2          = 06;
IOCFG1          = 2E;
IOCFG0          = 00;
FIFOTHR = 47;
//Tamaño del paquete y palabra de sincronía
SYNC1           = 33;
SYNC0           = CC;
PKTLEN  = 43;
//Resolución por broadcast, CRC, dirección de nodo, número de canal
PKTCTRL1        = C3;
PKTCTRL0        = 00;
ADDR            = 0F;
CHANNR = 00;
//Frecuencia If, compensación de frecuencia, frecuencia carrier
FSCTRL1 = 06;
FSCTRL0 = 00;
FREQ2           = 10;
FREQ1           = A7;
FREQ0           = 60;
//Separación del canal, ancho de banda del filtro Rx, datarate, codificación, modulación, criterio aceptación de sincronía,
MDMCFG4         = 9A;
MDMCFG3         = 83;
MDMCFG2         = 0A;
MDMCFG1         = 72;
MDMCFG0         = 93;
DEVIATN= 42;
//control de la máquina de estados
MCSM2           = 07;
MCSM1           = 20;
MCSM0           = 18;
//Ganancias y muestreos de los filtros de corrección de desviación para el PLL
FOCCFG  = 16;
BSCFG           = 6C;
AGCCTRL2        = 03;
AGCCTRL1        = 40;
AGCCTRL0        = 91;
//Configuraciones sobre la función de Wake-up on Radio event
WOREVT1                 = 87;
WOREVT0                 = 6B;
WORCTRL                 = FB;
//Front end de TX y RX
FREND1          = 56;
FREND0          = 10;
//Calibración del sintetizador
FSCAL3          = EF;
FSCAL2          = 0B;
FSCAL1          = 3C;
FSCAL0          = 1F;
// Configuracion del oscilador alternativo interno
RCCTRL1         = 00;
RCCTRL0         = 00;
//Prueba y debugging
FSTEST          = 00;
PTEST           = 7F;
AGCTEST                 = 00;
TEST2           = 81;
TEST1           = 35;
TEST0           = 09;
PATABLE0        = C0;
```

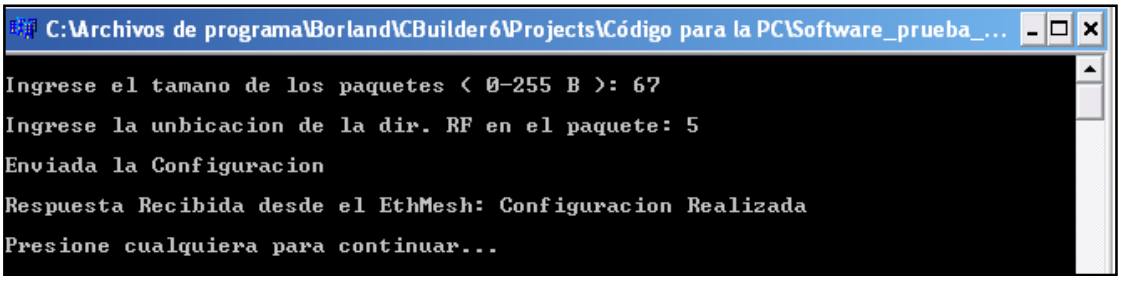
Anexo B.11 Menú principal de configuración y respuesta luego de enviar a cargar configuraciones.



Anexo B.12 Menú principal de configuración y respuesta luego de enviar a cargar configuraciones (continuación).



Anexo B.13 Menú de configuración del tamaño de paquete y posición del campo dirección RF destino.



Anexo B.14 Menú principal de enlace con la red inalámbrica.